

“十二五”普通高等教育本科国家级规划教材

模式识别与智能计算 ——MATLAB 技术实现 (第3版)

杨淑莹 张桦 著

電子工業出版社·

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书广泛吸取统计学、神经网络、数据挖掘、机器学习、人工智能、群智能计算等学科的先进思想和理论,将其应用到模式识别领域中;以一种新的体系,系统、全面地介绍模式识别的理论、方法及应用。全书分为14章,内容包括:模式识别概述,特征的选择与优化,模式相似性测度,基于概率统计的贝叶斯分类器设计,判别函数分类器设计,神经网络分类器设计(BP神经网络、径向基函数神经网络、自组织竞争神经网络、概率神经网络、对向传播神经网络、反馈型神经网络),决策树分类器设计,粗糙集分类器设计,聚类分析,模糊聚类分析,禁忌搜索算法聚类分析,遗传算法聚类分析,蚁群算法聚类分析,粒子群算法聚类分析。

本书内容新颖,实用性强,理论与实际应用密切结合,以手写数字识别为应用实例,介绍理论运用于实践的的实现步骤及相应的Matlab代码,为广大研究工作者和工程技术人员对相关理论的应用提供借鉴。

本书可作为高等院校计算机工程、信息工程、生物医学工程、智能机器人学、工业自动化、模式识别等学科本科生、研究生的教材或教学参考书,也可供相关工程技术人员参考。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有,侵权必究。

图书在版编目(CIP)数据

模式识别与智能计算:MATLAB 技术实现/杨淑莹,张桦著. —3版. —北京:电子工业出版社,2015.4
ISBN 978-7-121-25790-2

I. ①模… II. ①杨… ②张… III. ①模式识别-计算机辅助计算-Matlab 软件 ②人工智能-计算机辅助计算-Matlab 软件 IV. ①O235-39 ②TP183

中国版本图书馆 CIP 数据核字(2015)第 066200 号

责任编辑:张 榕

印 刷:

装 订:

出版发行:电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本:787×1092 1/16 印张:23.5 字数:601.6 千字

版 次:2008 年 1 月第 1 版

2011 年 7 月第 2 版

2015 年 4 月第 3 版

印 次:2015 年 4 月第 1 次印刷

印 数:3 000 册 定价:58.00 元

凡所购买电子工业出版社的图书,如有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010) 88254888。

质量投诉请发邮件至 zlt@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线:(010) 88258888。

第3版前言

模式识别已经成为当代高科技研究的重要领域之一,它已发展成为一门独立的新学科。模式识别技术迅速扩展,已经应用在人工智能、机器人、系统控制、遥感数据分析、生物医学工程、军事目标识别等领域,几乎遍及各个学科领域,在国民经济、国防建设、社会发展的各个方面得到广泛应用,产生了深远的影响。

第3版书以实用性、可操作性和实践性为宗旨,精简内容,去掉没有实践性代码的理论内容,书中所述理论知识均提供实现步骤、示范性代码及验证实例的效果图示,以达到理论与实践相结合的目的。为使读者更好地理解相关知识,叙述内容更加详实,补充了概率神经网络知识等新内容。同时,将群体智能的先进思想扩充到模式识别体系中,以一种新的体系,系统、全面地介绍了模式识别的理论、方法及应用。全书分为三部分(共14章)。第一部分基础篇,内容包括模式识别的基本概念,特征的选择与提取,模式相似性测度。这一部分介绍模式识别的基本概念和基本方法。第二部分分类器设计篇,内容包括:贝叶斯(Bayes)分类器设计,判别函数设计,神经网络分类器设计,决策树分类器设计,粗糙集分类器设计。这一部分利用手写数字分类识别的具体实例把模式识别方法结合起来,为广大研究工作者和工程技术人员对相关理论的应用起到借鉴作用。第三部分聚类分析,内容包括基本聚类算法,模拟退火聚类分析,模糊聚类分析,禁忌搜索算法聚类分析,遗传算法聚类分析,群体智能聚类算法(蚁群算法聚类分析,粒子算法群聚类分析)。这一部分采用一幅含有需要聚类分析的图像形象生动地说明各种聚类算法。

国内外论述模式识别技术的书籍不少,但由于这一领域涉及深奥的数学理论,往往使实际工作者感到困难,而大部分书是罗列模式识别的各种算法,见不到算法的实际效果和各种算法对比的结果,而这正是学习者和实际工作者所需要了解和掌握的内容。目前还确实缺少一本关于模式识别技术在实际应用方面具有系统性、可比性和实用性的参考书。

本书特点如下:

1. 选用新技术。除了介绍许多重要经典的内容以外,书中还包括了最近十几年来才刚刚发展起来的并被实践证明有用的新技术、新理论,比如支持向量机、BP神经网络、RBF神经网络、PNN神经网络、CPN神经网络、SORNN神经网络、决策树、粗糙集理论、模糊集理论、模拟退火、遗传算法、蚁群算法、粒子群算法等,并将这些新技术应用于模式识别当中,提供这些新技术的实现方法和源代码。

2. 实用性强。针对实例介绍理论和技术,使理论和实践相结合,避免了空洞的理论说教。书中实例取材于手写数字模式识别,对于数字识别属于多类问题,在实际应用中具有广泛的代表性,读者对程序稍加改进,就可以应用到不同的场合,如文字识别、字符识别、图形识别等。

3. 编排合理,符合认知规律。针对每一种模式识别技术,书中分为理论基础、实现步骤、编程代码三部分。在掌握了基本理论之后,按照实现步骤的指导,可以了解算法的实现思路和方法,再进一步体会短小精悍的核心代码,学习者可以很快掌握模式识别技术,经过应用本书

提供的实例程序,立刻会见到算法的实际效果。书中所有算法都用 Matlab 编程实现,便于读者学习和应用。

本书内容基本涵盖了目前“模式识别”重要的理论和方法,但并没有简单地将各种理论方法堆砌起来,而是将作者自身的研究成果和实践经验传授给读者,在介绍各种理论和方法时,将不同算法应用于实际中,内容包括需要应用模式识别技术解决的问题,模式识别理论的讲解和推理,将理论转化为编程的步骤,计算机能够运行的源代码,计算机运行模式识别算法程序后的效果,以及不同算法应用于同一个问题的效果对比。使读者面对如此丰富的理论和方法不至于无所适从,而是有所学就会有所用。

由于至今还没有统一的、有效的可应用于所有的模式识别的理论,当前的一种普遍看法是,不存在对所有的模式识别问题都适用的单一模型和解决识别问题的单一技术,我们所要做的是把模式识别方法与具体问题结合起来,把模式识别与统计学、神经网络、数据挖掘、机器学习、人工智能、群智能计算等学科的先进思想和理论结合起来,为读者提供一个多种理论的测试平台,并在此基础上,深入掌握各种理论的效能和应用的可能性,互相取长补短,开创模式识别应用的新局面。

本书可作为高等院校计算机工程、信息工程、生物医学工程、智能机器人学、工业自动化、模式识别等学科研究生、本科生的教材或教学参考书,也可供有关工程技术人员参考。

参加本书编写的还有祁稳杰、申婷婷、叶诚、谷林、陈品、郭俊、姜娇娇、祁家家、杨作寿、马振洋、邓飞、陶先贵等,他们在作者指导下的研究工作中付出了辛苦的劳动,取得了有益的研究成果,正是在他们的努力下本书得以顺利完成,在此表示衷心的感谢。同时,对张桦教授、徐伯夏研究员、李兰友教授给予的帮助和支持表示衷心的感谢。本书的出版得到天津理工大学出版基金的资助。由于编者业务水平和实践经验有限,书中缺点与错误在所难免,欢迎读者予以指正!

作者将不辜负广大读者的期望,努力工作,不断充实新的内容。为方便广大读者,提供了技术支持电子邮箱:ysying1262@126.com。读者可通过该邮箱及时与作者取得联系,获得技术支持。

著 者

目 录

第 1 章 模式识别概述	1
1.1 模式识别的基本概念	1
1.2 模式识别的基本方法	4
1.3 统计模式识别	6
1.3.1 统计模式识别研究的主要问题	6
1.3.2 统计模式识别方法简介	7
1.4 分类分析	10
1.4.1 分类器设计	10
1.4.2 判别函数	12
1.4.3 分类器的选择	15
1.4.4 训练与学习	15
1.5 聚类分析	16
1.5.1 聚类的设计	17
1.5.2 基于试探法的聚类设计	18
1.5.3 基于群体智能优化算法的聚类设计	19
1.6 模式识别的应用	25
本章小结	26
习题 1	26
第 2 章 特征的选择与优化	27
2.1 特征空间优化设计问题	27
2.2 样本特征库初步分析	28
2.3 样品筛选处理	29
2.4 特征筛选处理	29
2.5 特征评估	31
2.6 基于主成分分析的特征提取	33
2.7 特征空间描述与分析	36
2.7.1 特征空间描述	36
2.7.2 特征空间分布分析	41
2.8 手写数字特征提取与分析	44
2.8.1 手写数字特征提取	44
2.8.2 手写数字特征空间分布分析	46
本章小结	50

习题 2	50
第 3 章 模式相似性测度	51
3.1 模式相似性测度的基本概念	51
3.2 距离测度分类法	54
3.2.1 模板匹配法	54
3.2.2 基于 PCA 的模板匹配法	56
3.2.3 基于类中心的欧式距离法分类	58
3.2.4 马氏距离分类	60
3.2.5 夹角余弦距离分类	62
3.2.6 二值化的夹角余弦距离法分类	63
3.2.7 二值化的 Tanimoto 测度分类	64
本章小结	66
习题 3	66
第 4 章 基于概率统计的贝叶斯分类器设计	67
4.1 贝叶斯决策的基本概念	67
4.1.1 贝叶斯决策所讨论的问题	67
4.1.2 贝叶斯公式	68
4.2 基于最小错误率的贝叶斯决策	70
4.3 基于最小风险的贝叶斯决策	73
4.4 贝叶斯决策比较	75
4.5 基于二值数据的贝叶斯分类实现	76
4.6 基于最小错误率的贝叶斯分类实现	79
4.7 基于最小风险的贝叶斯分类实现	82
本章小结	85
习题 4	86
第 5 章 判别函数分类器设计	87
5.1 判别函数的基本概念	87
5.2 线性判别函数	88
5.3 线性判别函数的实现	92
5.4 感知器算法	93
5.5 增量校正算法	100
5.6 LMSE 验证可分性	106
5.7 LMSE 分类算法	112
5.8 Fisher 分类	115
5.9 基于核的 Fisher 分类	118
5.10 势函数法	125
5.11 支持向量机	130
本章小结	136

习题 5	137
第 6 章 神经网络分类器设计	138
6.1 神经网络的基本原理	138
6.1.1 人工神经元	138
6.1.2 人工神经网络模型	141
6.1.3 神经网络的学习过程	144
6.1.4 神经网络在模式识别问题上的优势	144
6.2 BP 神经网络	145
6.2.1 BP 神经网络的基本概念	145
6.2.2 BP 神经网络分类器设计	151
6.3 径向基函数神经网络(RBF)	161
6.3.1 径向基函数神经网络的基本概念	161
6.3.2 径向基函数神经网络分类器设计	166
6.4 自组织竞争神经网络	168
6.4.1 自组织竞争神经网络的基本概念	169
6.4.2 自组织竞争神经网络分类器设计	171
6.5 概率神经网络(PNN)	174
6.5.1 概率神经网络的基本概念	174
6.5.2 概率神经网络分类器设计	177
6.6 对向传播神经网络(CPN)	180
6.6.1 对向传播神经网络的基本概念	180
6.6.2 对向传播神经网络分类器设计	182
6.7 反馈型神经网络(Hopfield)	186
6.7.1 Hopfield 网络的基本概念	186
6.7.2 Hopfield 神经网络分类器设计	189
本章小结	191
习题 6	191
第 7 章 决策树分类器设计	192
7.1 决策树的基本概念	192
7.2 决策树分类器设计	193
本章小结	200
习题 7	200
第 8 章 粗糙集分类器设计	201
8.1 粗糙集理论的基本概念	201
8.2 粗糙集在模式识别中的应用	206
8.3 粗糙集分类器设计	210
本章小结	223
习题 8	224

第 9 章 聚类分析	225
9.1 聚类的设计	225
9.2 基于试探的未知类别聚类算法	229
9.2.1 最临近规则的试探法	229
9.2.2 最大最小距离算法	233
9.3 层次聚类算法	235
9.3.1 最短距离法	236
9.3.2 最长距离法	239
9.3.3 中间距离法	243
9.3.4 重心法	246
9.3.5 类平均距离法	250
9.4 动态聚类算法	254
9.4.1 K 均值算法	254
9.4.2 迭代自组织的数据分析算法 (ISODATA)	258
9.5 模拟退火聚类算法	263
9.5.1 模拟退火的基本概念	263
9.5.2 基于模拟退火思想的改进 K 均值聚类算法	266
本章小结	273
习题 9	273
第 10 章 模糊聚类分析	274
10.1 模糊集的基本概念	274
10.2 模糊集运算	276
10.2.1 模糊子集运算	276
10.2.2 模糊集运算性质	278
10.3 模糊关系	278
10.4 模糊集在模式识别中的应用	283
10.5 基于模糊的聚类分析	284
本章小结	298
习题 10	298
第 11 章 禁忌搜索算法聚类分析	299
11.1 禁忌搜索算法的基本原理	299
11.2 禁忌搜索的关键参数和相关操作	301
11.3 基于禁忌搜索算法的聚类分析	304
本章小结	313
习题 11	313
第 12 章 遗传算法聚类分析	314
12.1 遗传算法的基本原理	314
12.2 遗传算法的构成要素	316

12.2.1 染色体的编码	316
12.2.2 适应度函数	317
12.2.3 遗传算子	318
12.3 控制参数的选择	320
12.4 基于遗传算法的聚类分析	321
本章小结	333
习题 12	333
第 13 章 蚁群算法聚类分析	334
13.1 蚁群算法的基本原理	334
13.2 聚类数目已知的蚁群聚类算法	337
13.3 聚类数目未知的蚁群聚类算法	346
本章小结	351
习题 13	351
第 14 章 粒子群算法聚类分析	352
14.1 粒子群算法的基本原理	352
14.2 基于粒子群算法的聚类分析	355
本章小结	360
习题 14	361
参考文献	362

第 1 章 模式识别概述

本章要点:

- ☑ 模式识别的基本概念
- ☑ 模式识别的基本方法
- ☑ 统计模式识别
- ☑ 分类器设计
- ☑ 聚类设计
- ☑ 模式识别的应用

1.1 模式识别的基本概念

模式识别(Pattern Recognition)就是机器识别、计算机识别或机器自动识别,目的在于让机器自动识别事物。例如手写数字的识别,结果就是将手写的数字分到具体的数字类别中;智能交通管理系统的识别,就是判断是否有汽车闯红灯,闯红灯的汽车车牌号码;还有文字识别,语音识别,图像中物体识别,等等。该学科研究的内容是使机器能做以前只能由人类才能做的事,具备人所具有的对各种事物与现象进行分析、描述与判断的部分能力。模式识别是直观的、无所不在的,实际上人类在日常生活的每个环节,都从事着模式识别的活动。人和动物较容易做到的模式识别,但对计算机来说却是非常困难的。让机器能识别、分类,就需要研究识别的方法,这就是这门学科的任务。

模式识别是信号处理与人工智能的一个重要分支。人工智能是专门研究用机器人模拟人的动作、感觉和思维过程与规律的一门科学,而模式识别则是利用计算机专门对物理量及其变化过程进行描述与分类,通常用来对图像、文字、相片以及声音等信息进行处理、分类和识别。它所研究的理论和方法在很多科学和技术领域中得到了广泛的重视与应用,推动了人工智能系统的发展,扩大了计算机应用的可能性。模式识别诞生于 20 世纪 20 年代,随着 20 世纪 40 年代计算机的出现,50 年代人工智能的兴起,模式识别在 60 年代初迅速发展为一门科学。其研究的目的是利用计算机对物理对象进行分类,在错误概率最小的条件下,使识别的结果尽量与客观物体相符合。让机器辨别事物的最基本方法是计算,原则上讲是对计算机要分析的事物与标准模板的相似程度进行计算。例如,要识别一个手写的数字,就要将它与从 0 到 9 的模板做比较,看跟哪个模板最相似,或最接近。因此首先要能从度量中看出不同事物之间的差异,才能分辨当前要识别的事物。因此,最关键的是找到有效地度量不同类别事物差异的方法。

在模式识别学科中,就“模式”与“模式类”而言,模式类是一类事物的代表,而“模式”则是某一事物的具体体现,如数字 0、1、2、3、4、5、6、7、8、9 是模式类,而用户任意手写的一个数字或任意一个印刷数字则是“模式”,是数字的具体化。广义上说,模式(pattern)是供模仿用的完美无缺的标本,通常,把通过对具体的个别事物进行观察所得到的具有时间和空间分布的信息称之为模式,而把模式所属的类别或同一类别中模式的总体称为模式类。模式识别(Pattern

Recognition)是指对表征事物或现象的各种形式的(数值的、文字的和逻辑关系的)信息进行处理和分析,以对事物或现象进行描述、辨认、分类和解释的过程,是信息科学和人工智能的重要组成部分。

1. 模式的描述方法

在模式识别技术中,被观测的每个对象称为样品,例如,在手写数字识别中每个手写数字可以作为一个样品,如果共写了 N 个数字,我们把这 N 个数字叫做 N 个样品($X_1, X_2, \dots, X_j, \dots, X_N$),其中 0 有 N_0 个样品,1 有 N_1 个样品,2 有 N_2 个样品,3 有 N_3 个样品, \dots , 一共有 $\omega_1, \omega_2, \dots, \omega_M$ ($M=10$) 个不同的类别。

对于一个样品来说,必须确定一些与识别有关的因素,作为研究的根据,每一个因素称为一个特征。模式就是样品所具有的特征的描述。模式的特征集又可写成处于同一个特征空间的特征向量,特征向量的每个元素称为特征,该向量也因此称为特征向量。一般我们用小写英文字母 x, y, z 来表示特征。如果一个样品 X 有 n 个特征,则可把 X 看作一个 n 维列向量,该向量 X 称为特征向量,记作:

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = (x_1, x_2, \dots, x_n)^T$$

若有一批样品共有 N 个,每个样品有 n 个特征,这些数值可以构成一个 n 行 N 列的矩阵,称为原始资料矩阵,见表 1-1。

表 1-1 原始资料矩阵

样 品 特 征	X_1	X_2	\dots	X_j	\dots	X_N
x_1	x_{11}	x_{21}	\dots	x_{j1}	\dots	x_{N1}
x_2	x_{12}	x_{22}	\dots	x_{j2}	\dots	x_{N2}
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
x_i	x_{1i}	x_{2i}	\dots	x_{ji}	\dots	x_{Ni}
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
x_n	x_{1n}	x_{2n}	\dots	x_{jn}	\dots	x_{Nn}

模式识别问题就是根据 X 的 n 个特征来判别模式 X 属于 $\omega_1, \omega_2, \dots, \omega_M$ 类中的哪一类。待识别的不同模式都在同一特征空间中考察,不同模式由于性质上的不同,它们在各特征取值范围上有所不同,因而会在特征空间的不同区域中出现。要记住向量的运算是建立在各个分量基础之上的。因此,模式识别系统的目标是在特征空间和解释空间之间找到一种映射关系。特征空间由从模式得到的对分类有用的度量、属性或基元构成的空间。解释空间由 M 个所属类别的集合构成。

如果一个对象的特征观察值为 $\{x_1, x_2, \dots, x_n\}$, 它可构成一个 n 维的特征向量值 X , 即 $X = (x_1, x_2, \dots, x_n)^T$, 式中, x_1, x_2, \dots, x_n 为特征向量 X 的各个分量。一个模式可以看作 n 维空间中

的向量或点,此空间称为模式的特征空间 R_n 。在模式识别过程中,要对许多具体对象进行测量,以获得更多观测值。其中有均值、方差、协方差与协方差矩阵等。

2. 模式识别系统

一个典型的模式识别系统如图 1-1 所示,由数据获取、预处理、特征提取选择、分类决策及分类器设计五部分组成。一般分为上下两部分。上半部完成未知类别模式的分类;下部分属于分类器设计的训练过程,利用样品进行训练,确定分类器的具体参数,完成分类器的设计。而分类决策在识别过程中起作用,对待识别的样品进行分类决策。

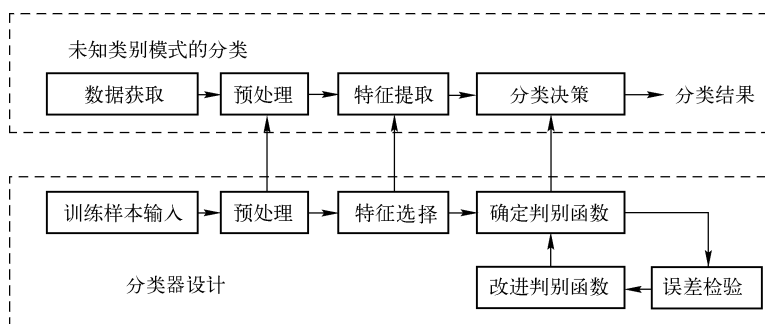


图 1-1 典型的模式识别系统

在设计模式识别系统时,需要注意模式类的定义、应用场合、模式表示、特征提取和选择、聚类分析、分类器的设计和学习、训练和测试样本的选取、性能评价等。针对不同的应用目的,模式识别系统各部分的内容可以有很大的差异,特别是在数据处理和模式分类这两部分,为了提高识别结果的可靠性往往需要加入知识库(规则)以对可能产生的错误进行修正,或通过引入限制条件大大缩小待识别模式在模型库中的搜索空间,以减少匹配计算量。在某些具体应用中,如机器视觉,除了要给出被识别对象是什么物体外,还要求出该物体所处的位置和姿态以引导机器人的工作。下面分别简单介绍模式识别系统的工作原理。

模式识别系统组成单元功能如下所述。

(1) 数据获取:是指利用各种传感器把被研究对象的各种信息转换为计算机可以接受的数值或符号(串)集合。习惯上,称这种数值或符号(串)所组成的空间为模式空间。这一步的关键是传感器的选取。为了从这些数字或符号(串)中抽取对识别有效的信息,必须进行数据处理,包括数字滤波和特征提取。所获取的数据要用计算机可以运算的符号来表示所研究的对象,一般获取的数据类型有:

二维图像:文字、指纹、地图、照片等;

一维波形:脑电图、心电图、季节震动波形等;

物理参量和逻辑值:体温、化验数据、参量正常与否的描述。

(2) 预处理:是为了消除输入数据或信息中的噪声,排除不相干的信号,只留下与被研究对象的性质和采用的识别方法密切相关的特征(如表征物体的形状、周长、面积等)。举例来说,在进行指纹识别时,指纹扫描设备每次输出的指纹图像会随着图像的对比度、亮度或背景等的不同而不同,有时可能还会变形,而人们感兴趣的仅仅是图像中的指纹线、指纹分叉点、端点等,而不需要指纹的其他部分或背景。因此,需要采用合适的滤波算法,如基于块方图的方

向滤波、二值滤波等,过滤掉指纹图像中这些不必要的部分。需要对输入测量仪器或其他因素所造成的退化现象进行复原、去噪声,提取有用信息。

(3) 特征提取和选择:是指从滤波数据中衍生出有用的信息,从许多特征中寻找出最有效的特征,将维数较高的测量空间(原始数据组成的空间)转变为维数较低的特征空间(分类识别赖以进行的空间)。以降低后续处理过程的难度。通过特征选择和提取形成模式的特征空间。人类很容易获取的特征,对于机器来说就很难获取了,特征选择和提取是模式识别的一个关键问题。一般情况下,候选特征种类越多,得到的结果应该越好。但是,由此可能会引发维数灾害,即特征维数过高,计算机难以求解。因此,数据处理阶段的关键是滤波算法和特征提取方法的选取。不同的应用场合,采用的滤波算法和特征提取方法,以及提取出来的特征也会不同。

(4) 分类决策:在特征空间中用模式识别方法把被识别对象归为某一类别。该阶段最后输出的可能是对象所属的类型,也可能是模型数据库中与被识别对象最相似的模式编号。

(5) 分类器设计:模式分类或描述通常是基于已经得到分类或描述的模式集合而进行的。人们称这个模式集合为训练集,由此产生的学习策略称为监督学习。学习也可以是非监督性学习,在此意义下产生的系统不需要提供模式类的先验知识,而是基于模式的统计规律或模式的相似性学习判断模式的类别。基本做法是在样品训练集基础上确定判别函数,改进判别函数和误差检验。

研究模式识别的主要目的是如何用计算机进行模式识别,对样本进行分类。执行模式识别的计算机系统称为模式识别系统。设计人员按需要设计模式识别系统,而该系统被用来执行模式分类的具体任务。

1.2 模式识别的基本方法

模式识别方法(Pattern Recognition Method)是一种借助于计算机对信息进行处理、判决分类的数学统计方法。应用模式识别方法的首要步骤是建立模式空间。所谓模式空间是指在考察一客观现象时,影响目标的众多指标构成的多维空间。模式识别就是对多维空间中各种模式的分布特点进行分析,对模式空间进行划分,识别各种模式的聚类情况,从而做出判断或决策。分析方法就是利用“映射”和“逆映射”技术。映射是指将多维模式空间通过数学变换到二维平面,多维空间的所有模式(样本点)都投影在该平面内。在二维平面内,不同类别的模式分布在不同的区域之间有较明显的分界域。由此确定优化方向返回到多维空间(原始空间),得出真实信息,帮助人们找出规律或做出决策,指导实际工作或实验研究。

在 d 维特征空间已经确定的前提下,讨论分类器设计问题,其实是一个选择什么准则、使用什么方法,并将已确定的 d 维特征空间划分成决策域的问题。针对不同的对象和不同的目的,可以用不同的模式识别理论或方法,目前基本的技术方法有统计模式识别、句法模式识别。

1. 统计模式识别

统计模式识别方法是发展较早也比较成熟的一种方法。被识别对象首先数字化,变换为适于计算机处理的数字信息。一个模式常常要用很大的信息量来表示。许多模式识别系统在数字化环节之后还要进行预处理,用于除去混入的干扰信息并减少某些变形和失真。随后再进行特征抽取,即从数字化后或预处理后的输入模式中抽取一组特征,模式可用特征空间中的

一个点或一个特征矢量表示。所谓特征是选定的一种度量,它对于一般的变形和失真保持不变或几乎不变,并且只含尽可能少的冗余信息。特征抽取过程将输入模式从对象空间映射到特征空间。这种映射不仅压缩了信息量,而且易于分类。在决策理论方法中,特征抽取占有重要的地位,但尚无通用的理论指导,只能通过分析具体识别对象决定选取何种特征。特征抽取后可进行分类,即从特征空间再映射到决策空间。为此引入鉴别函数,由特征矢量计算出相应的各类别的鉴别函数值,通过鉴别函数值的比较实行分类。

统计模式识别方法适用于给定的有限数量样本集,其基本思想是将特征提取阶段得到的特征向量定义在一个特征空间中,这个空间包含了所有的特征矢量。不同的特征向量,或者说不同类别的对象,都对应于此空间中的一点。在分类阶段,则利用统计决策的原理对特征空间进行划分,从而达到识别不同特征对象的目的。已知研究对象统计模型或已知判别函数类条件下,根据一定的准则通过学习算法能够把 d 维特征空间划分为 c 个区域,每一个区域与每一类别相对应,模式识别系统在进行工作时只要判断被识别的对象落入哪一个区域,就能确定出它所属的类别。统计识别中应用的统计决策分类理论相对比较成熟,研究的重点是特征提取。基于统计模式识别方法有多种方法,通常较为有效,现已形成了完整的体系。尽管方法很多,但从根本上讲,都是直接利用各类的分布特征,即利用各类的概率分布函数、后验概率或隐含地利用上述概念进行分类识别。其中基本的技术为聚类分析、判别类域界面法、统计判决等。

(1) 聚类分析

在聚类分析中,利用待分类模式之间的“相似性”进行分类,更相似的作为一类,更不相似的作为另外一类。在分类过程中不断地计算所分划的各类的中心,下一个待分类模式以其与各类中心的距离作为分类的准则。聚类准则的确定,基本上有两种方式。一种是试探方式,即凭直观和经验,针对实际问题定义一种相似性测度的阈值,然后按最近邻规则指定某些模式样本属于某一聚类类别。例如欧氏距离测度,它反映样本间的近邻性,但将一个样本分到两个类别中的一个时,必须规定一距离测度的阈值作为聚类的判别准则,按最近邻规则的简单试探法和最大最小聚类算法就是采用这种方式。另一种是聚类准则函数法,即规定一种准则函数,其函数值与样品的划分有关。当取得极小值时,就认为得到了最佳划分。实际工作中采用最多的聚类方法之一是系统聚类法。它将模式样本按距离准则逐步聚类,类别由多到少,直到满足合适的分类要求为止。

(2) 判别类域界面法

判别类域界面法中,用已知类别的训练样本产生判别函数,这相当于学习或训练。根据待分类模式代入判别函数后所得值的正负而确定其类别。判别函数提供了相邻两类判决域的界面,最简单、最实用的判别函数是线性判别函数。利用线性判别函数进行决策就是用一个超平面对特征空间进行分割。超平面的方向由权向量决定,而位置由阈值权的数值确定,超平面把特征空间分割为两个决策区域。

(3) 统计判决

在统计判决中,在一些分类识别准则下严格地按照概率统计理论导出各种判决准则,这些判决准则要用到各类的概率密度函数、先验概率或条件概率,即贝叶斯法则。

2. 句法结构模式识别

句法识别是对统计识别方法的补充。统计方法用数值来描述图像的特征,句法方法则用符号来描述图像特征。它模仿了语言学中句法的层次结构,采用分层描述的方法,其基本思想是把一个模式描述为较简单的子模式的组合,子模式又可描述为更简单的子模式的组合,最终得到一个树形的结构描述,在底层的最简单的子模式称为模式基元。在句法方法中选取基元的问题相当于在决策理论方法中选取特征的问题。通常要求所选的基元能对模式提供一个紧凑的反映其结构关系的描述,又要易于用非句法方法加以抽取。显然,基元本身不应该含有重要的结构信息。模式以一组基元和它们的组合关系来描述,称为模式描述语句,这相当于在语言中,句子和短语用词组合,词用字符组合一样。基元组合成模式的规则,由所谓语法来指定。一旦基元被鉴别,识别过程可通过句法分析进行,即分析给定的模式语句是否符合指定的语法,满足某类语法的即被分入该类。

句法结构模式识别又称结构方法或语言学方法,主要用于文字识别、遥感图形的识别与分析,以及纹理图像的分析。该方法的特点是识别方便,能够反映模式的结构特征,能够描述模式的性质,对图像畸变的抗干扰能力较强。如何选择机缘是本方法的一个关键问题,尤其是当存在干扰及噪声时,抽取基元更困难,且易失误。把复杂图像分解为单层或多层的简单子图像,主要突出了识别对象的结构信息。图像识别是从统计方法发展起来的,而句法方法扩大了识别的能力,使其不局限于对象物的分类,而且还用于景物的分析和物体结构的识别。

模式识别方法的选择取决于问题的性质。如果被识别的对象极为复杂,而且包含丰富的结构信息,一般采用句法方法,当被识别对象不很复杂或不含明显的结构信息,一般采用决策理论方法。统计方法发展较早,比较成熟,取得了不少应用成果,能考虑干扰、噪声等影响,识别模式基元能力强;但是它对结构复杂的模式抽取特征困难,不能反映模式的结构特征,难以描述模式的性质,对模式本身的结构关系很少利用,难以从整体角度考虑识别问题。而很多识别问题,并不是用简单的分类就能解决的,更重要的是要弄清楚这些模式的结构关系。句法结构模式识别能反映模式的结构特性,识别方便,可从简单的基元开始,由简至繁。描述模式的性质;另一方面,单纯的句法模式识别方法没有考虑到模式所受到的环境、噪声的干扰等不稳定因素的影响,当存在干扰及噪声时,抽取基元困难,且易失误。

在应用中,常常将这两种方法结合起来,分别施加于不同的层次,会收到较好的效果。两者的结合已是模式识别问题的一个研究方向,在这方面,提出了随机文法、属性文法等一些新的研究方向,并取得了一定的成果。

1.3 统计模式识别

1.3.1 统计模式识别研究的主要问题

统计模式识别研究的主要问题有:特征的选择与优化、分类判别及聚类判别。

(1) 特征的选择与优化

如何确定合适的特征空间是设计模式识别系统一个十分重要的问题,对特征空间进行优化有两种基本方法,一是特征选择,如果所选用的特征空间能使同类物体分布具有紧致性,可

为分类器设计成功提供良好的基础;反之,如果不同类别的样品在该特征空间中混杂在一起,再好的设计方法也无法提高分类器的准确性。另一种是特征的组合优化,通过一种映射变换改造原特征空间,构造一个新的精简的特征空间。

(2) 分类判别

已知若干个样品的类别及特征,例如,手写阿拉伯数字的判别是具有10类的分类问题,机器首先要知道每个手写数字的形状特征,对同一个数字,不同的人有不同的写法,甚至同一个人对同一个数字也有多种写法,就必须让机器知道它属于哪一类。因此对分类问题需要建立样品库。根据这些样品库建立判别分类函数,这一过程是由机器来实现的,称为学习过程,然后对一个未知的新对象分析它的特征,决定它属于哪一类。这是一种监督分类的方法。

(3) 聚类判别

已知若干对象和它们的特征,但不知道每个对象属于哪一个类,而且事先并不知道究竟分成多少类,用某种相似性度量的方法,即“人以类聚,物以群分”,把特征相同的归为一类。例如,手写了若干个阿拉伯数字,把相同的数字归为一类。这是一种非监督学习的方法。

机器识别也往往借鉴人的思维活动,像人类一样找出待识别物的外形或颜色等特征,进行分析、判断,然后加以分门别类,即识别它们。模式识别的方法很多,很难将其全部概括,也很难说那种方法最佳,常常需要根据实际的情况运用多种方法进行实验,然后选择最佳的分类方法。

1.3.2 统计模式识别方法简介

基于统计模式识别方法有多种方法:例如模板匹配法、判别函数法、神经网络分类法、基于规则推理法等。这些方法各有特点及应用范围,它们不能相互取代,只能共存,相互促进、借鉴、渗透。一个较完善的识别系统很可能是综合利用上述各类识别方法的观点、概念和技术而形成的。

1. 模板匹配

模板匹配的原理是选择已知的对象作为模板,与待测物体进行比较,从而识别目标。将待分类样品与标准模板进行比较,看跟哪个模板匹配程度更好些,从而确定待测试样品的分类。而近邻法则在原理上属于模板匹配。它将训练样品集中的每个样品都作为模板,用测试样品与每个模板做比较,看与哪个模板最相似(即为近邻),就按最近似的模板的类别作为自己的类别。譬如A类有10个训练样品,因此有10个模板,B类有8个训练样品,就有8个模板。任何一个待测试样品在分类时与这18个模板都算一算相似度,如最相似的那个近邻是B类中的一个,就确定待测试样品为B类,否则为A类。因此原理上说近邻法是最简单的。但是近邻法有一个明显的缺点就是计算量大,存储量大,要存储的模板很多,每个测试样品要对每个模板计算一次相似度,因此在模板数量很大时,计算量也是很大的。模板匹配的另一个缺点是由于匹配的点很多,理论上最终可以达到最优解,但在实际中却很难做到。模板匹配主要应用于对图像中对象物位置的检测,运动物体的跟踪,不同光谱或者不同摄影时间所得的图像之间位置的配准等。模板匹配的计算量很大,相应的数据的存储量也很大,而且随着图像模板的增大,运算量和存储量以几何级数增长。如果图像和模板大到一定程度,就会导致计算机无法

处理,随之也就失去了图像识别的意义。

2. 判别函数

设计判别函数的形式有两种方法:基于概率统计的分类法和几何分类法。

(1) 基于概率统计的分类器

基于概率统计的分类器主要有基于最小错误率的贝叶斯决策、基于最小风险的贝叶斯决策。

直接使用贝叶斯决策需要首先得到有关样品总体分布的知识,包括各类先验概率 $P(\omega_i)$ 及类条件概率密度函数,计算出样品的后验概率 $P(\omega_i | X)$,并以此作为产生判别函数的必要数据,设计出相应的判别函数与决策面。当各类样品近似于正态分布时,可以算出使错误率最小或风险最小的分界面,以及相应的分界面方程。因此如果能从训练样品估计出各类样品服从近似的正态分布,可以按贝叶斯决策方法对分类器进行设计。

这种利用训练样品的方法是通过它的概率分布进行估计,然后用它进行分类器设计,这种方法则称为参数判别方法。它的前提是对特征空间中的各类样品的分布已很清楚,一旦要测试分类样品的特征向量值 X 已知,就可以确定 X 对各类的后验概率,也就可按相应的准则计算与分类。所以判别函数等的确定取决于样品统计分布的有关知识。因此参数分类判别方法一般只能用在有统计知识的场合,或能利用训练样品估计出参数的场合。

贝叶斯分类器可以用一般的形式给出数学上严格的分析证明:在给出某些变量的条件下,能使分类所造成的平均损失最小,或者分类决策的风险最小。因此能计算出分类器的极限性能。贝叶斯决策采用分类器中最重要的指标——错误率作为产生判别函数和决策面的依据,因此它给出了最一般情况下适用的“最优”分类器设计方法,对各种不同的分类器设计技术在理论上都有指导意义。

(2) 判别函数分类法

由于一个模式通过某种变换映射为一个特征向量后,该特征向量可以理解为特征空间的一个点,在特征空间中,属于一个类的点集,总是在某种程度上与属于另一个类的点集相分离,各个类之间确定可分的,因此如果能够找到一个判别函数(线性或非线性函数),把不同类的点集分开,则分类任务就解决了。判别分类器不依赖于条件概率密度的知识,可以理解为通过几何的方法,把特征空间分解为对应于不同类别的子空间。而且呈线性的分离函数,将使计算简化。分离函数又分为线性判别函数和非线性判别函数。

3. 神经网络分类

人工神经网络的研究起源于对生物神经系统的研究。它将若干处理单元(即神经元)通过一定的互连模型连结成一个网络,这个网络通过一定的机制可以模仿人的神经系统的动作过程,以达到识别分类的目的。人工神经网络区别于其他识别方法的最大特点是它对待识别的对象不要求有太多的分析与了解,具有一定的智能化处理的特点。神经网络侧重于模拟和实现人认知过程中的感知觉过程、形象思维、分布式记忆、自学习和自组织过程,与符号处理是一种互补的关系。但神经网络具有大规模并行、分布式存储和处理、自组织、自适应和自学习的能力,特别适用于处理需要同时考虑许多因素和条件的、不精确和模糊的信息处理问题。

神经网络可以看成是从输入空间到输出空间的一个非线性映射,它通过调整权重和阈值来“学习”或发现变量间的关系,实现对事务的分类。由于神经网络是一种对数据分布无任何要求的非线性技术,它能有效解决非正态分布、非线性的评价问题,因而得到广泛的应用。由于神经网络具有信息的分布存储,并行处理及自学习能力等特点,它在泛化处理能力上显示出较高的优势,可处理一些环境信息十分复杂,背景知识不清楚,推理规则不明确的问题。允许样品有较大的缺损、畸变。缺点是目能识别的模式类还不够多,模型在不断丰富与完善中。

4. 基于规则推理法

基于规则推理法是对待识客体运用统计(或结构、模糊)识别技术,或人工智能技术,获得客体的符号性表达即知识性事实后,运用人工智能技术对知识的获取、表达、组织、推理方法,确定该客体所归属的模式类(进而使用)的方法。它是一种与统计模式识别、句法模式识别相并列(又相结合)的基于逻辑推理的智能模式识别方法。它主要包括知识表示、知识推理和知识获取三个环节。

通过样本训练集构建推理规则进行模式分类的方法主要有:决策树和粗糙集理论。决策树学习是以实例为基础的归纳学习算法。它着眼于从一组无次序、无规则的实例中推理出决策树表示形式的分类规则。决策树整体为一棵倒长的树,分类时,采用自顶向下的递归方式,在决策树的内部结点进行属性值的比较并根据不同属性判断从该结点向下的分支,在决策树的叶结点得到结论。粗糙集理论反映了认知过程在非确定、非模型信息处理方面的机制和特点,是一种有效的非单调推理工具。粗糙集以等价关系为基础,用上、下近似两个集合来逼近任意一个集合,该集合的边界区域被定义为上近似集和下近似集之差集,边界区域就是那些无法归属的个体。上、下近似两个集合可以通过等价关系给出确定的描述,边界域的元素数目可以被计算出来。这两个理论在数据的决策和分析、模式识别、机器学习与知识发展等方面有着成功的应用,已成为信息科学最活跃的研究领域之一。

基于规则推理法适用于已建立了关于知识表示与组织、目标搜索及匹配的完整体系。对需通过众多规则的推理达到识别确认的问题,有很好的效果。但是当样品有缺损,背景不清晰,规则不明确甚至有歧义时,效果不好。

5. 模糊模式识别法

模糊模式识别的理论基础是模糊数学。它根据人辨识事物的思维逻辑,吸取人脑的识别特点,将计算机中常用的二值逻辑转向连续逻辑。模糊识别的结果是用被识别对象隶属于某一类别的程度即隶属度来表示的,一个对象可以在某种程度上属于某一类别,而在另一种程度上属于另一类别。一般常规识别方法则要求一个对象只能属于某一类别。基于模糊集理论的识别方法有:最大隶属原则识别法、择近原则识别法和模糊聚类法。由于用隶属度函数作为样品与模块间相似程度的度量,故往往能反映它们整体的和主要的特性,从而允许样品有相当程度的干扰与畸变。但准确合理的隶属度函数往往难以建立,故限制了它的应用。伴随着各门学科,尤其是人文、社会学科及其他“软科学”的不断发展,数学化、量化的趋势也开始在这些领域中显现。模糊模式识别不再简单局限于自然科学的应用,同时也被应用到社会科学,特别是经济管理学科方面。

6. 支持向量机的模式识别

支持向量机(Support Vector Machine, SVM)方法是求解模式识别和函数估计问题的有效工具,是由 Vapnik 领导的 AT&Bell 实验室研究小组在 1963 年提出的一种新的非常有潜力的分类技术,其基本思想是:先在样本空间或特征空间,构造出最优超平面,使得超平面与不同类样本集之间的距离最大,从而达到最大的泛化能力。支持向量机结构简单,并且具有全局最优性和较好的泛化能力,自提出以来得到了广泛的研究。

支持向量机在数字图像处理方面的应用是寻找图像像素之间的特征的差别,即从像素点本身的特征和周围的环境(临近的像素点)出发,寻找差异,然后将各类像素点区分出来。

上述的各种方法各有特点及应用范围,它们不能相互取代,可以取长补短,互相补充、促进、借鉴、渗透。一个较完善的识别系统很可能是综合利用上述各类识别方法的观点、概念和技术而形成的。

1.4 分类分析

模式识别分类问题是指根据待识别对象所呈现的观察值,将其分到某个类别中去。具体步骤是建立特征空间中的训练集,已知训练集里每个点所属类别,从这些条件出发,寻求某种判别函数或判别准则,设计判决函数模型,然后根据训练集中的样品确定模型中的参数,便可将这模型用于判别,利用判别函数或判别准则去判别每个未知类别的点应该属于哪一个类。

1.4.1 分类器设计

如何做出合理的判决就是模式识别分类器要讨论的问题。在统计模式识别中,感兴趣的主要问题并不是决策正误,而在于如何使决策错误造成的分类误差在整个识别过程中的风险代价达到最小。模式识别算法的设计都是强调“最佳”与“最优”,即希望所设计的系统在性能上最优。这种最优是指对某一种设计原则讲的,这种原则称为准则,常用的准则有最小错误率准则、最小风险准则、近邻准则、Fisher 准则、均方误差最小准则、感知准则等。设计准则,并使该准则达到最优的条件是设计模式识别系统最基本的方法。模式识别中以确定准则函数来实现优化的计算框架。分类器设计使用什么原则是关键,会影响到分类器的效果。不同的决策规则反映了分类器设计者的不同考虑,对决策结果有不同的影响。分类决策在识别过程中起作用,对待识别的样品进行分类决策。

一般说来, M 类不同的物体应该具有各不相同的属性值,在 n 维特征空间中,各自有不同的分布。当某一特征向量值 X 只为某一类物体所特有,对其做出决策是容易的,也不会出什么差错。问题在于出现模棱两可的情况时,由于属于不同类的待识别对象存在着呈现相同特征值的可能,即所观察到的某一样品的特征向量为 X ,而在 M 类中又有不止一类可能呈现这一 X 值。如图 1-2 所示, A 、 B 直线之间的样品属于不同类别,但是他们具有相同的特征值。例如癌症病人初期症状与正常人的症状相同,其两类别样品分别用“-”与“+”表示。从图中可见这两类样品在二维特征空间中相互穿插,这两类样品很难用简单的分界线将它们完全分开。如果用一直线作为分界线,称为线性分类器,对图中所示的样品分布情况,无论直线参数如何设计,总会有错分类发生。此时,任何决策都存在判错的可能性。

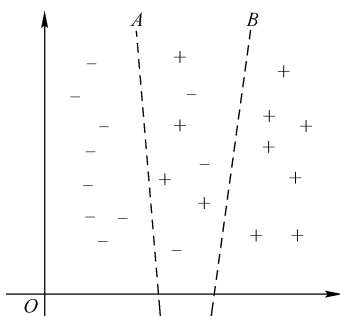


图 1-2 分界线示意图(改后)

模式识别的基本计算框架——制定准则函数,实现准则函数极值化。常用的准则有以下几种:

(1) 最小错分率准则

完全以减少分类错误为原则,这是一个通用原则,如图 1-2 所示,如果以错分类最小为原则分类,则图中 A 直线可能是最佳的分界线,它使错分类的样品数量为最小。

(2) 最小风险准则

当接触到实际问题时,可以发现使错误率最小并不一定是一个普遍适用的最佳选择。有的分类系统将错分率多少看成最重要的指标,如对语音识别、文字识别来说这是最重要的指标。而有的分类系统对于错分率多少并不看重,而是要考虑错分类的不同后果,如对医疗诊断、地震、天气预报等,例如可能多次将没有发生的地震预报成有地震,也有可能将发生的地震预报为没有地震,这类系统并不看重错分率,而是要考虑错分类引起的严重后果。例如上面讨论过的细胞分类中,把正常细胞错分为癌细胞,或相反方向的错误,其严重性是截然不同的。以 B 直线划分,有可能把正常细胞误判为异常细胞,“+”样品错分成“-”类,给人带来不必要的痛苦,错分率多;但以 A 直线划分,有可能把癌细胞误判为正常细胞,“-”分成“+”类,会使病人因失去及早治疗的机会而遭受极大的损失,但错分率少。为使总的损失为最小,那么 B 直线就可能比 A 直线更适合作为分界线。这是基于最小风险的原理。

由此可见,根据不同性质的错误会引起不同程度的损失这一考虑出发,我们宁可扩大一些总的错误率,但也要使总的损失减少。因此引入风险、损失这些概念,以便在决策时兼顾不同后果的影响。在实际问题中计算损失与风险是复杂的,在使用数学式子计算时,往往用赋予不同权值来表示。在做出决策时,要考虑所承担的风险。基于最小风险的贝叶斯决策规则正是为了体现这一点而产生的。

(3) 近邻准则

近邻准则是分段线性判别函数的一种典型方法。这种方法主要依据同类物体在特征空间具有聚类特性的原理。同类物体由于其性质相近,它们在特征空间中应具有聚类的现象,因此可以利用这种性质产生分类决策的规则。例如有两类样品,可以求出每一类的平均值,对于任何一个未知样品,先求出它到各个类的平均值距离,判断距离谁近就属于谁。

(4) Fisher 准则

根据两类样品一般类内密集、类间分离的特点,寻找线性分类器最佳的法线向量方向,使

两类样品在该方向上的投影满足类内尽可能密集,类间尽可能分开。把它们投影到任意一根直线上,有可能不同类别的样品就混在一起了,无法区分。如图 1-3(a)所示,样品投影到 x_1 或 x_2 轴无法区分。若把直线绕原点转动一下,就有可能找到一个方向,样品投影到这个方向的直线上,各类样品就能很好地分开,如图 1-3(b)所示。因此直线方向选择是很重要的。一般来说,总能够找到一个最好的方向,使样品投影到这个方向的直线上很容易分开。如何找到这个最好的直线方向,以及如何实现向最好方向投影的变换,这正是 Fisher 算法要解决的基本问题。

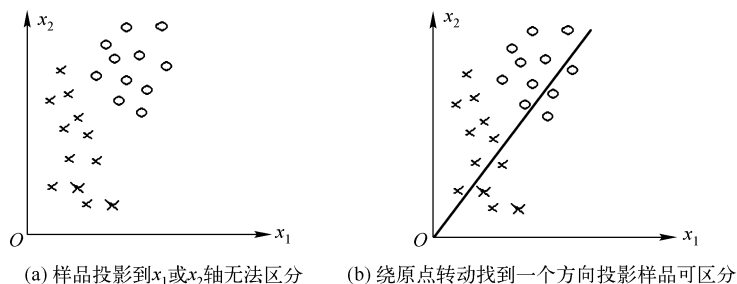


图 1-3 Fisher 线性判别原理示意图

这说明如果两类分布围绕各自均值的确相近, Fisher 准则可使错误率较小, Fisher 方法实际上涉及到维数压缩的问题。

(5) 感知准则

感知准则函数以使错分类样品到分界面距离之和最小为原则。提出利用错误提供信息实现迭代修正的学习原理,即利用错分类提供的信息修正错误。这种思想对机器学习的发展以及人工神经网络的发生发展产生深远影响。其优点是通过错分类样品提供的信息对分类器函数进行修正,这种准则是人工神经网络多层感知器的基础。

(6) 最小均方误差准则

LMSE 算法以最小均方误差作为准则。

1.4.2 判别函数

无论是应用概率统计的分类法还是应用几何分类法,最终都转化为确定判别函数形式。

1. 二类情况

对于只有简单的两类情况,判别函数形式如图 1-4 所示,根据计算结果的符号将 X 分类。

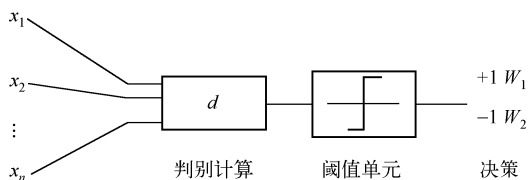


图 1-4 两类分类器形式(改后)

这里首先假定判别函数 $d(X)$ 是 X 的线性函数, $d(X) = W^T X + W_0$ 。

(1) 二维特征

在二维模式空间中存在一线性判别函数:

$$d(X) = w_1 x_1 + w_2 x_2 + w_3 = 0 \quad (1-1)$$

(2) n 维特征

用矢量 $X = (x_1, x_2, \dots, x_n)^T$ 来表示模式, 一般的线性判别函数形式为:

$$d(X) = w_1 x_1 + w_2 x_2 + \dots + w_n x_n + w_{n+1} = W_0^T X + w_{n+1} \quad (1-2)$$

式中 $W_0 = (w_1, w_2, \dots, w_n)^T$ 称为权矢量或参数矢量。如果在所有模式矢量的最末元素后再附加元素 1, 则式(1-7)可以写成

$$d(X) = W^T X \quad (1-3)$$

$$W = (w_1, w_2, \dots, w_n, w_{n+1})$$

2. 多类情况

对于多类别问题, 假设有 M 类模式 $\omega_1, \omega_2, \dots, \omega_M$, 对于 n 维空间中的 M 个类别, 就要给出 M 个判别函数: $d_1(X), d_2(X), \dots, d_M(X)$, 各个判别函数构成分类器基本形式如图 1-5 所示。若 X 属于第 i 类, 则有:

$$d_i(X) > d_j(X) \quad j = 1, 2, \dots, M, i \neq j \quad (1-4)$$

特殊情况, 有:

$$d_i(X) = d_j(X) \quad (1-5)$$

这时, 是在两类的分界线上, X 即属于第 i 类, 也属于第 j 类, 因此这种判别无效, 还必须考虑其他特征, 重新判别。

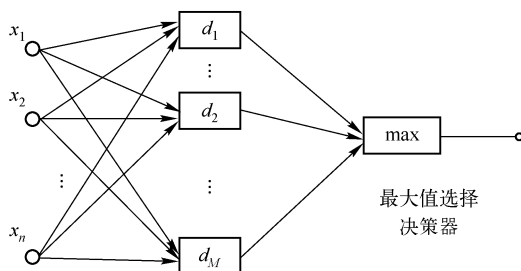


图 1-5 判别函数构成的多类分类器形式(改后)

判别函数的自变量是待测样品 X 的 n 个特征值, 将待测样品 X 的 n 个特征值分别代入 M 个判别函数中, 计算出各个函数表达式的结果, 哪一个最大, 待测样品 X 就属于哪一个类。 M 个判别函数一般表示成 $d_i(X)$, 如果 $d_i(X) = \max_j d_j(X)$, 则称特征空间的这一点 X 是第 i 类的决策域。由 $d_i(X)$ 占主导地位的区域称为第 i 类的决策域, 将它表示成 R_i , 如果第 i 类决策域 R_i 与第 j 类决策域相邻, 则它们之间有边界。在边界上有 $d_i(X) = d_j(X)$, 该式是一个方程式, 称为决策面方程。决策面是一种统称, 当特征空间只是一维时, 一个决策面实际上只是一个点。在二维特征空间里, 决策面是一条曲线。三维则是一个曲面, 超过三维的空间, 决策面是一个超曲面。判别函数 $d_i(X)$ 用于表达决策规则的某些函数。判别函数 $d_i(X)$ 与决策面方程

$d_i(X) = d_j(X)$ 是密切相关的,并且都是由相应决策规则所确定的。

对于线性情况,判别函数形式为:

$$\begin{aligned} d(X) &= w_1x_1 + w_2x_2 + \cdots + w_nx_n + w_{n+1} \\ &= W_0^T X + w_{n+1} = W^T X \end{aligned} \quad (1-6)$$

其中, $X = (x_1, x_2, \cdots, x_n, 1)^T$, $W = (w_1, w_2, \cdots, w_{n+1})^T$

对于非线性情况,判别函数形式为:

$$\begin{aligned} d(X) &= w_1f_1(x) + w_2f_2(x) + \cdots + w_nf_n(x) + w_{n+1} \\ &= \sum_{i=1}^{M+1} w_if_i(x), i = 1, 2, \cdots, M \\ d(X) &= w_1f_1(x) + w_2f_2(x) + \cdots + w_nf_n(x) + w_{n+1} \\ &= \sum_{i=1}^{n+1} w_if_i(x) \end{aligned} \quad (1-7)$$

3. 参数的确定

由于决策域的分界面是用数学式子来描述的,如线性函数,或各种非线性函数等。因此确定分界面方程包括选择函数类型与确定最佳参数两个部分。一般说来,选择函数类型是由设计者确定的,但其参数的确定则是通过一个学习过程来实现的,是一个迭代实现优化的过程。

由此可见设计分类器,一是选定所用的判别函数类型,二是确定方程的两个参数(权向量 W , 阈值 w_0)。对于线性判别函数来说,方程的形式固定为线性,维数固定为特征向量的维数,方程组的数量取决于待识别对象的类数。既然方程组的数量、维数和形式已定,则对判别函数的设计就是确定函数的各系数,即线性方程的各个权值。确定线性方程的各个权值的方法有 Fisher 准则、感知器算法、增量校正算法、LMSE 算法等。

线性分类器设计任务是在给定样品集和集合内各样品所属类别条件下,确定线性判别函数的各项系数,对待测样品进行分类时,能满足相应的准则函数 J 为最优的要求。这种方法的具体过程可大致分为:

(1) 确定使用的判别函数类型或决策面方程类型,如线性分类器、分段线性分类器、非线性分类器等或近邻法等;

(2) 按需要确定一准则函数 J : 如 Fisher 准则、感知器算法、增量校正算法、LMSE 算法。增量校正算法与感知器算法的实现相似,只不过是在进行权矢量修正时加上了权系数;LMSE 算法以最小均方误差作为准则。

(3) 确定准则函数 J 达到极值时 W^* 及 W_0^* 的具体数值,从而确定判别函数,完成分类器设计。在计算机上确定各权值时采用的是“训练”或“学习”的方法,就是挑选一批已分类的样品,把这批样品输入到计算机的“训练”程序中去,通过多次迭代,最后准则函数 J 达到极值,得到正确的线性判别函数。

这种方法绕过统计分布状况的分析,绕过参数估计这一环,而企图对特征空间实行划分,称为非参数判别分类法,即不依赖统计参数的分类法。非参数判别分类方法的核心是由训练样品集提供的信息直接确定决策域的划分方法。线性判别函数法是一类较为简单的判别函数,计算量少,它以模式的样品集线性可分离为前提。

1.4.3 分类器的选择

在讨论了判别函数等概念后,设计分类器的任务就清楚了。根据样品分布情况来确定分类器的类型。在设计分类器的方法时,要有一个样品集,样品集中的样品用一个各分量含义已经确定的向量来描述,也就是说对要分类的样品怎样描述这个问题是已经确定的。在这种条件下研究用贝叶斯分类器、线性分类器与非线性分类器等,以及这些分类器的其他设计问题。

按照基于统计参数的决策分类方法,判别函数及决策面方程的类别确定是由样品分布规律决定的,贝叶斯决策是基于统计分布确定的情况下计算的,如果要按贝叶斯决策方法设计分类器,就必须设法获得必需的统计参数。当有条件得到准确的统计分布知识,具体说来包括各类先验概率 $P(\omega_i)$ 及类条件概率密度函数,从而可以计算出样品的后验概率 $P(\omega_i | X)$,并以此作为产生判别函数的必要依据,利用贝叶斯决策来实现对样品的分类。但是,在这些参数未知的情况下使用贝叶斯决策方法,就得有一个学习阶段。在这个阶段,设法获得一定数量的样品,然后从这些样品数据获得对样品概率分布的估计。有了概率分布的估计后,才能对未知的新样品按贝叶斯决策方法实行分类。

在一般情况下要得到准确的统计分布知识是极其困难的事。当实际问题中并不具备获取准确统计分布的条件时,使用几何分类器。几何分类器设计过程主要是判别函数、决策面方程的确定过程。设计分类器首先要确定准则函数,然后再利用训练样品集确定该分类器的参数,以求使所确定的准则达到最佳。在使用分类器时,样品的分类由其判别函数值决定。判别函数可以是线性函数、也可以设计成非线性函数。设特征向量的特征分量数目为 n ,可分类数目为 M ,符合某种条件就可使用线性分类器,正态分布条件下一般适合用二次函数决策面。

(1) 若可分类数目 $M = 2(n + 1) \approx 2n$,则几乎无法用一个线性函数分类器将它们分成两类。

(2) 在模式识别中,理论上, $M > n + 1$ 的线性分类器不能应用,但是如果一个类别的特征向量在空间中密集地聚集在一起,几乎不和其他类别的特征向量混合在一起,则无论 M 多大,线性分类器的效果总是良好的。在字符识别机中,线性函数分类器已经证明能够提供良好的识别效果,它能识别数量很大的字符识别任务。

因此,在手写数字识别中,只要读者规范书写数字,不同数字类别的特征空间可以看成彼此分离的,而同一类别的数字在特征空间中集群性质较好,应用线性分类器是可行的。相反,如果特征向量的类集群性质不好,则线性分类器的效果总是不理想,此时,必须求助于非线性分类器。

1.4.4 训练与学习

所谓模式识别中的学习与训练是从训练样品提供的数据中找出某种数学式子的最优解,这个最优解使分类器得到一组参数,按这种参数设计的分类器使人们设计的某种准则达到极值。确定分类决策的具体数学公式是通过分类器设计这个过程确定的。在模式识别学科中一般把这个过程称为训练与学习的过程。

分类的规则是依据训练样品提供信息确定的。分类器设计在训练过程中完成,利用一批

训练样品,包括各种类别的样品,由这些样品大致勾画出各类事物在特征空间分布的规律性,为确定使用什么样的数学公式及这些公式中的参数提供信息。一般来说,决定使用什么类型的分类函数是人为决定的。分类器参数的选择或在学习过程得到的结果取决于设计者选择什么样的准则函数。不同准则函数的最优解对应不同的学习结果,得到性能不同的分类器。数学式子中的参数则往往通过学习来确定,分类器有一种学习过程,如果发现当前采用的分类函数会造成分类错误,那么利用错误提供应如何纠正的信息,就可以使分类函数朝正确的方向前进,这就形成了一种迭代的过程,如果分类函数及其参数使出错的情况越来越少,就可以说是逐渐收敛,学习过程就收到了效果,设计也就可以结束。

训练与学习的过程常常用到以下三个概念:

(1) 训练集:是一个已知样品集,在监督学习方法中,用它来开发模式分类器。

在分类实例中,样品库训练集是程序开发人员按照自己的手写数字习惯来书写的数字,因此,会造成对读者手写的数字分类有误的情况,为了尽量避免此类情况发生,我们把每次添加的手写数字放在样品训练集的首位,读者可以尽量多写一些数字以使程序适应其书写式样。

(2) 测试集:在设计识别和分类系统时没有用过的独立样品集。

在分类实例中,以读者自己手写的数字作为测试检验,每写一个可以用各种模式识别算法进行检验。这样的好处是在相同的样品特征值下,可以对不同的模式识别算法进行比较,找出最佳适应算法。

(3) 系统评价原则:就是判断该模式识别系统能否正确分类,为了更好地对模式识别系统性能进行评价,必须使用一组独立于训练集的测试集对系统进行测试。

1.5 聚类分析

前面介绍的分类问题是利用已知类别的样品(训练集)来构造分类器的。其训练集样品是已知类别的,所以又称为有监督学习(或有教师分类)。在已知类别样品的“指导”(监督)下对单个待测样品进行分类。聚类问题则不同,它是事先不了解一批样品中的每一个样品的类别或其他的先验知识,而唯一的分类根据是样品的特性。利用样品的特性来构造分类器。这种分类称为无监督分类,通常称为聚类或集群。

聚类分析是指事先不了解一批样品中的每一个样品的类别或其他的先验知识,而唯一的分类根据是样品的特征,利用某种相似性度量的方法,把特征相同或相近的归为一类,实现聚类划分。

聚类分析就是对探测数据进行分类分析的一个工具,许多学科要根据所测得的或感知到的相似性对数据进行分类,把探测数据归入到各个聚合类中,且在同一个聚合类中的模式比不同聚合类中的模式更相似,从而对模式间的相互关系做出估计。聚类分析的结果可以被用来对数据提出初始假设,分类新数据,测试数据的同类型及压缩数据。

聚类算法的重点是寻找特征相似的聚合类。人类是二维的最佳分类器,然而大多数实际的问题涉及到高维的聚类。对高维空间内的数据的直观解释,其困难是明显的,另外,数据也不会服从规则的理想分布,这就是有大量聚类算法出现在文献中的原因。在图像中进行聚类分析,一幅图像中含有多个物体,需要对不同的物体分割标识。

1.5.1 聚类的设计

1. 聚类的定义

Everitt 提出一个聚合类是一些相似的实体集合,而且不同聚合类的实体是不相似的。在一个聚合类内的两个点间的距离小于在这个类内任一点和不在这个类内的另一任一点间的距离。聚合类可以被描述成在 n 维空间内存在较高密度点的连续区域和较低密度点的区域,而较低密度点的区域把其他较高密度点的区域分开。

在模式空间 S 若给定 N 个样品 X_1, X_2, \dots, X_N , 聚类的定义是:按照相互类似的程度找到相应的区域 R_1, R_2, \dots, R_M , 对任意 $X_i (i=1, 2, \dots, N)$ 归入其中一类,而且不会同时属于两类,即:

$$\begin{aligned} R_1 \cup R_2 \cup \dots \cup R_M &= R \\ R_i \cap R_j &= \emptyset \quad (i \neq j) \end{aligned}$$

这里 \cup 、 \cap 分别为并集,交集。

选择聚类的方法应以一个理想的聚类概念为基础。然而,如果数据不满足由聚类技术所做的假设,则算法不是去发现真实的结构而是在数据上强加上某一种结构。

2. 聚类准则

设有未知类别的 N 个样品,要把它们划分到 M 类中去,可以有多种优劣不同的聚类方法,怎样评价聚类的优劣,这就需要确定一种聚类准则。但客观地说,聚类的优劣是就某一种评价准则而言的,很难有对各种准则均呈优良表现的聚类方法。

聚类准则的确定,基本上有两种方法。一种是试探法,根据所分类的问题,试探性进行样品的划分,确定一种准则,并用它来判断样品分类是否合理。例如,以距离函数作为相似性的度量,用不断修改的阈值,来探究对此种准则的满足程度,当取得极小值时,就认为得到了最佳划分。另一种是群体智能方法,随着对生物学的深入研究,人们逐渐发现自然界中个体行为简单、能力非常有限,但当它们一起协同工作时,表现出并不是简单的个体能力的叠加,而是非常复杂的行为特征,群体智能优化算法在没有集中控制并且不提供全局模型的前提下,利用群体的优势,分布搜索,这种方法一般能够比传统的优化方法更快地发现复杂优化问题的最优解,为寻找复杂问题的最佳方案提供了新的思路和新方法。

下面给出一种简单而又广泛应用的准则,即误差平方和准则:

设有 N 个样品,分属于 $\omega_1, \omega_2, \dots, \omega_M$ 类,设有 N_i 个样品的 ω_i 类,其均值为

$$m_i = \frac{1}{N_i} \sum_{X \in \omega_i} X \quad \rightarrow \quad \overline{X^{(\omega_i)}} = \frac{1}{N_i} \sum_{X \in \omega_i} X \quad (1-8)$$

因为有若干种方法可将 N 个样品划分到 M 类中去,因此对应一种划分,可求得一个误差平方和 J ,要找到使 J 值最小的那种划分。定义误差平方和

$$J = \sum_{i=1}^M \sum_{X \in \omega_i} |X - m_i|^2 \quad \rightarrow \quad J = \sum_{i=1}^M \sum_{X \in \omega_i} |X - \overline{X^{(\omega_i)}}|^2 \quad (1-9)$$

经验表明,当各类样品均很密集,各类样品个数相差不大,而类间距离较大时,适合采用误差平方和准则,如图 1-6(a) 所示。若各类样品数相差很大,类间距离较小时,就有可能将样品数多的类一分为二,而得到的 J 值却比大类保持完整时小,误以为得到了最优划分,实际上得

到了错误的划分,如图 1-6(b)所示。

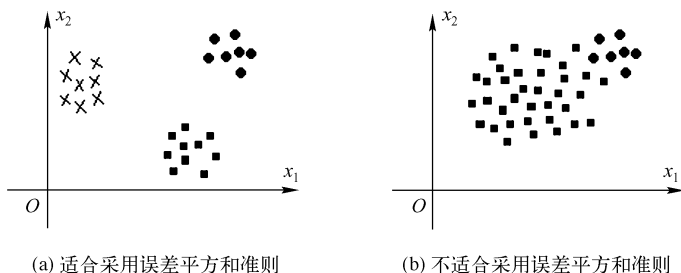


图 1-6 样品分布与误差平方和准则关系

1.5.2 基于试探法的聚类设计

基于试探法的聚类设计采用假设某种分类方案,确定一种聚类准则,计算 J 值,找到 J 值最小的那一种分类方案,则认为该方法为最优分类。基于试探的未知类别聚类算法,包括最临近规则的试探法和最大最小距离算法。

1. 最临近规则的试探法

假设前 i 个样品已经被分到 k 个类中。则第 $i+1$ 个样品应该归入哪一个类中? 假设归入 ω_a 类,要使 J 最小,则应满足第 $i+1$ 个样品到 ω_a 类的距离小于给定的阈值,若大于给定的阈值 T ,则应为其建立一个新的类 ω_{k+1} 。在未将所有的样品分类前,类数是不能确定的。

这种算法与第一个中心的选取、阈值 T 的大小、样品排列次序及样品分布的几何特性有关。这种方法运算简单,当有关于模式几何分布的先验知识作指导给出阈值 T 及初始点时,则能较快地获得合理的聚类结果。

2. 最大最小距离试探法

最临近规则的试探法受到阈值 T 的影响很大。阈值的选取是聚类成败的关键之一。最大最小距离算法充分利用样品内部特性,计算出所有样品间的最大距离作为归类阈值的参考,改善了分类的准确性。例如,采用某样品到某一个聚类中心的距离小于最大距离的一半,则归入该类,否则建立新的聚类中心。

3. 层次聚类试探

层次聚类方法对给定的数据集进行层次的分解,直到某种条件满足为止。具体又可分为合并、分裂两种方案。

合并的层次聚类是一种自底向上的策略,首先将每个对象作为一个类,然后根据类间距离的不同,合并距离小于阈值的类,合并一些相似的样品,直到终结条件被满足,合并算法会在每一步减小聚类中心数量,聚类产生的结果来自于前一步的两个聚类的合并;绝大多数层次聚类方法属于这一类,它们只是在相似度的定义上有所不同。

分裂的层次聚类与合并的层次聚类相反,采用自顶向下的策略,它首先将所有对象置于同

一个簇中,然后逐渐细分为越来越小的样品簇,直到达到了某个终止条件。分裂算法与合并算法的原理相反,在每一步增加聚类中心数量,每一步聚类产生的结果,都是将前一步的一个聚类中心分裂成两个得到的。

1.5.3 基于群体智能优化算法的聚类设计

群体智能算法的仿生计算一般由初始化种群、个体更新和群体更新三个过程组成。下面分别介绍这三个过程的仿生计算机制。

1. 初始化种群

在任何一种群体智能算法中,都包含种群的初始化。种群的初始化是假设每个样品已经被随机分到某个类中,产生若干个个体,人为地认为这些群体中的每一个个体为所求问题的解。因此,一般需要对所求问题的解空间进行编码操作,将具体的实际问题以某种解的形式给出,便于对问题的描述和求解。对于初始化种群的产生一般有两种方式:一种是完全随机产生的方法,另一种是结合先验知识产生初始种群。在没有任何先验知识的情况下往往采用第一种方式,而第二种方式可以使得算法较快的收敛到最优解。种群的初始化主要包括问题解形式的确定、算法参数的选取、评估函数的确定等。

(1) 问题解形式的确定

对于任何一类优化问题,在应用群体智能算法求解之前都需要对问题的解空间进行编码操作,将具体问题以一定的形式给出。不同的群体智能算法所对应的问题解的形式有所不同,在传统进化计算算法中,问题的解通常都是以染色体的形式给出;粒子群算法中问题的解是用粒子所经历的位置来表示;而在蜂群算法中,往往是通过蜜源来代表所求优化问题的可行解。对于各种解形式的编码方式一般有二进制编码、十进制编码、浮点数编码等,根据具体问题选择合适的编码方式可以加快算法的收敛速度。

(2) 算法参数的选取

合理选取算法的参数对于算法的求解有着重要的作用,好的参数值能够提高算法的准确性。在群体智能算法中,有关算法参数的选取,最为关键的是种群的规模和算法终止条件中关于最大迭代次数的确定。对于种群的规模也需要根据具体问题来确定。规模过大,将会增加算法的时间复杂度,降低了算法的效率;规模过小,又不容易使算法找到最优解,很容易使算法出现“早熟”现象。对于算法的最大迭代次数,需要根据多次实验来逐步确定,合理的选取最大迭代次数才可使算法收敛到全局最优解,并且提高执行效率。

不同的群体智能算法对应不同的控制参数。在传统进化计算算法中主要的控制参数还有交叉概率和变异概率。交叉概率用来控制两个个体之间信息的交互能力,变异概率用来控制产生新个体的能力,两种操作都增加了解的多样性。在传统遗传算法中,适应度值高的个体在一代中被选择的几率高;相应的浓度高、适应度值低的个体在一代中被选择的几率低。相应的浓度低,没有自我调节功能。而在免疫遗传算法中,除了抗体的适应度,还引入了免疫平衡算子,参与到抗体的选择中。免疫平衡算子对浓度高的抗体,进行抑制,反之浓度较低的抗体进行促进。根据抗体的适应度和浓度确定选择概率,它们的比例系数决定了适应度与浓度的作用大小。

粒子群算法中的主要参数为惯性权重和速度调节参数。惯性权重使得粒子保持运动惯性,速度调节参数表示粒子向自身最优和全局最优位置的加速项权重。在蚁群算法中,以前蚂蚁所留下的信息将会逐渐消失,比较重要的参数有信息素挥发系数,它直接影响算法的全局搜索能力及收敛速度。

猫群算法中的主要参数有分组率、记忆池大小、个体上每个基因的改变范围,由于自然界中的猫总是非常的懒散,经常花费大量的时间处在一种休息、张望的状态,称之为搜寻模式,一旦发现目标便进行跟踪,并且能够迅速的捕获到猎物,称之为跟踪模式,分组率控制了仿照真实世界中猫的行为模式。在蜂群算法中,为了保证蜜源的质量,将对蜜源的开采次数进行限制,开采次数过少不利于进行深入的局部搜索,开采次数过多容易造成蜜源枯竭,不利于跳出局部最优解。混合蛙跳算法采用模因分组算法模拟青蛙的聚群行为,模因组参数控制青蛙群体分成若干个小群体的数量。人工鱼群算法中的参数包含:尝试次数、感知范围、步长、拥挤度因子、人工鱼群数目等。细菌觅食算法中,趋化、繁殖、迁徙三种算子决定了算法的性能,相比其他算法,细菌觅食算法需要调节的参数较多:包括种群大小、前进步长、最大前进次数、趋化算子次数、繁殖算子次数、迁徙算子次数、迁徙概率,以及细菌觅食算法的优化能力和收敛速度与这些参数值的选择紧密相关。

在群体智能算法中,这些参数的选取都是在算法开始执行之前设定的,对于算法的性能和效率有很大的影响,如果参数选取不当,会使得算法的适应性变差甚至影响算法的整体性能。例如变异概率,如果取值太小就没有个体的更新机会,不容易产生新解,如果取值太大,容易造成个体发散;实践证明没有绝对的最优参数,针对不同的问题只有通过反复试验,才能选取较合适的参数,获得更好的收敛性能。

(3) 评估函数的确定

在所有的群体智能算法中,对于所求得的问题的解,都需要进行评价,可以帮助群体在迭代过程中选择出优良个体并且及早剔除较差个体,搜索出问题的最优解。在群体智能算法中,一般用适应度来评价个体(即问题的解)的好坏。对于适应度函数的确定,需要根据不同的问题进行设定。例如,在解决函数优化问题中,一般将目标函数直接作为适应度函数,通过求得它的值作为个体评价的标准;在手写数字识别问题中,我们用待识别数字与模板数字特征距离的倒数作为评估函数,此值越小说明特征越接近,其评价函数也就越高。

合理的选取评估函数对于算法的求解有着重要的作用,好的评估函数不但提高了评价的准确性,而且还会降低算法的时间复杂度,提高算法的执行效率。

2. 个体更新

个体的更新是群体智能算法中的关键一步,是群体质量提高的驱动力。在自然界中,个体的能力非常有限,行为也比较简单,但是,当多个简单的个体组合成一个群体之后,将会有非常强大的功能,能够许多完成复杂的工作。如蚁群能够完成筑巢、觅食,蜂群能高效完成采蜜、喂养、保卫工作,鱼群能够快速寻找食物、躲避攻击等工作。

群体智能算法中,采用简单的编码技术来表示一个个体所具有的复杂结构,在寻优搜索过程中,对一群用编码表示的个体进行简单的操作,本书将这些操作称作“算子”。个体的更新依靠这些算子实现,不同的群体智能算法仿生构造了不同的算子。如进化算法中的交叉算子(Crossover)、重组算子(Recombination),或者变异算子(Mutation)而繁殖出子代(Offsprings)、

选择算子(Selection);蚁群算法的蚂蚁移动算子、信息素更新算子。

个体更新的方式主要分为两种:一种是依靠自身的能力在解空间中寻找新的解;另一种是受到其他解(如当前群体中的最优解或邻域最优解)的影响更新自身。

(1) 依靠自身的能力进行局部搜寻

这类算法主要有:传统进化计算算法、Memetic 算法、猫群算法中的搜寻模式、蜂群算法中跟随蜂和侦查蜂的位置更新、细菌觅食算法、人工鱼群算法等。对于不同的算法,依靠自身的能力进行更新的方式又有所不同,下面介绍相关算法中个体更新的机制。

在传统的进化计算算法中,个体的位置更新主要有交叉、变异操作,交叉是模仿生物界的繁殖过程,对于完成选择操作的配对染色体以一定的概率通过某种方式进行互换部分基因,变异则是以一定的概率改变个体的基因来产生新的个体。

Memetic 算法的个体更新方式与传统进化计算算法稍有不同,即在完成遗传操作后,需要对群体中的所有个体进行局部搜索,使得解的质量进一步提高。局部搜索策略有很多,如爬山法、模拟退火法、禁忌搜索算法等,具体问题应选取合适的搜索策略以提高自身解的质量。

在猫群算法中当猫处于搜寻模式中,是通过自身位置进行复制的。对于复制出来的每一个副本,进行类似于进化计算算法的变异操作,之后进行适应度计算,选取最好的解来代替当前解。

蜂群算法中跟随蜂的位置更新是在引领蜂位置基础上加一个随机扰动实现的,而当蜜源枯竭时,即当前位置附近没有比引领蜂所在的蜜源更好的,则引领蜂更换角色,变为侦查蜂,侦查蜂随机地在解空间中产生一个新的解。

细菌觅食算法中的迁徙算子,满足迁徙概率的细菌执行迁徙操作,在整个解空间中随机地产生一个新解。

在人工鱼群算法中,在每条鱼尝试聚群算子和追尾算子后,若其适应度没有得到改善,则执行觅食算子,然而在尝试一定次数的觅食算子后,其适应度还是没有得到改善,此时人工鱼将在解空间中随机游到一个新的位置,作为一个新解。

我们看到如果仅仅是在原有解的基础上进行变异操作或加上一个随机扰动或做搜索操作,属于在原有解的附近做局部寻优,这种方式带来的个体位置更新范围不大;若直接将任意一个解赋给该个体,则具有较强的随机性,一定程度上增加了对于解空间的搜索范围,有利于求得全局最优解,但这也往往造成了一种盲目搜索。因此,在实际的问题求解中,我们要权衡两者的矛盾,以便更好地搜寻到全局最优解。

(2) 受到其他解的影响来更新自身

这类算法主要有:免疫算法、猫群算法中的跟踪模式、粒子群算法、混合蛙跳算法、蚁群算法、蜂群算法中引领蜂的更新。

免疫算法中,个体的更新过程是通过把上一代个体的优秀基因作为疫苗直接注射到下一代个体的相应基因位上,以此方式更新自身,使得下一代的个体优于上一代,不断地提高解的质量。

在猫群算法中当猫处于跟踪模式下,其位置更新并不是无目的的随机搜索,而是朝着最优解的方向不断逼近,通过猫所记忆的当前群体中的全局最优解来更新自身,提高了算法的搜索能力。

粒子群算法中个体的更新方式分为全局模式和局部模式,在全局模式中,粒子追随自身极值和全局极值,使得粒子向着最优解的方向前进,具有较快的收敛速度,但鲁棒性较差,而在局部模式中粒子只受自身极值和邻近粒子的影响,它具有较高的鲁棒性而收敛速度相对较慢。

在混合蛙跳算法中,是利用子群中处于最优位置的青蛙和处于全局最优位置的青蛙来更新蛙群中最差青蛙的位置,最差青蛙通过与这两者的交互使得自身位置不断更新,向着最优解靠拢。

在蚁群算法中,蚂蚁在觅食的过程中会根据先前蚂蚁在所行路径上留下的一种叫做“信息素”的东西来指引自己的路径,信息素越多表明该路径越好,使得后来的蚂蚁以较大的概率选择该路径,所有的蚂蚁都是通过这种特殊的消息交互机制实现个体的进化。

蜂群算法中跟随蜂搜索到的位置是以引领蜂为核心,在引领蜂所引领的位置上,加上引领蜂与群体中随机位置的偏差乘以随机数实现的,而引领蜂通过跟随蜂所搜寻到的最优位置实现自身更新。

通过与群体中其他个体的交流,使得自身的位置不断地得到更新,这种方式具有向优秀个体学习的机制,加快了算法的收敛速度和效率。

3. 群体更新

在基于群体概念的仿生智能算法中,群体更新是种群中个体更新的宏观表现,它对于算法的搜索和收敛性能具有重要作用。在不同的仿生群体智能优化算法中,存在着不同的群体更新方式。

(1) 个体更新实现群体更新

这种群体的更新主要依靠个体更新来实现。在仿生群体智能优化算法中,群体是由多个个体构成的,这些个体作为算法实行搜索的载体,代表了搜索问题的解空间,个体通过相同或不同的更新方式,改变自身的位置,使个体以较大概率得到改善。

而纵观不同的进化时代,群体更新代表了群体的流动方向。这种更新方式大多采用贪选择机制,即比较个体更新前后的评估值,保留较优的个体,使个体自身不断优化,趋向最优解,从而使群体的质量得到改善。

例如:在人工鱼群算法中,每条人工鱼通过一定规则尝试执行聚群行为、追尾行为、觅食行为,并选择改善最大的行为,进行位置的移动,实现个体更新。通过这种方式,使得每次迭代时,所有人工鱼都能以较大概率得到优化,从而实现整个种群质量的提高。

(2) 子群更新实现群体更新

在仿生群体智能优化算法中,有些算法将整个群体划分成多个子群,不同子群进行独立搜索,有些算法的每个子群运行相同的搜索模式,并实行子群间的协同合作与信息交互。例如在混合蛙跳算法中,算法将群体分为若干小群,在每一次迭代过程中,每个子群作为一个独立的进化单元,运行相同的搜索模式,子群在执行完一定次数的进化后,子群间的青蛙发生跳跃,子群体混合成新的群体,实现了种群的整体进化。

有些算法的每个子群则执行不同的搜索模式。例如,在猫群算法中,根据一定比例将整个猫群分为搜寻模式与跟踪模式两个子群,在每一次迭代过程中猫的行为模式都会进行重新随

机分配,不同子群运行不同的行为模式,在一定程度上提高了算法的全局搜索能力,使得整个群体解的质量不断地提高。

蜂群算法中,其群体的划分主要是根据蜜蜂工作职能的不同来区分的。在蜂群算法中,蜜蜂主要分为引领蜂、跟随蜂和侦查蜂,三种蜜蜂的位置更新方式不同。引领蜂是通过跟随蜂搜索到的最优位置进行更新的,而跟随蜂是在跟随的引领蜂位置附近进行搜索实现的,侦查蜂则是在解空间中的随机移动。这充分结合了全局搜索与局部搜索的特点与优点,并通过种子群功能的划分提高了算法的寻优性能。

这种划分子群的机制很大程度上提高了解的搜索范围,增加了解的多样性,虽然其群体更新方式较为复杂,但对于求解一些复杂的优化问题具有很好的效果。

(3) 选择机制实现群体更新

在前面介绍的两类算法中,个体更新后一般都能以较大概率取得更为优秀的新个体,而群体的更新主要通过选择算子来实现,若一味地选择适应度较高的个体,易造成种群内部多样性枯竭,使算法出现早熟,搜索陷入局部最优。这是种群内部个体多样性缺失的表现,需要其他方式来弥补这一缺点。由于即便是较差个体也保存着一些优秀基因,为避免产生退化现象,有必要让一些较差个体保留下来。

在进化计算体系中,个体更新后允许较差个体的出现,以此来扩充种群多样性,同时保证种群进化的整体方向。例如,在遗传算法中,父代个体通过交叉、变异等遗传算子产生子代个体,但是由于交叉、变异算子一般属于随机搜索,不能保证子代个体的质量,不可避免地会产生退化现象。遗传算法采用轮盘赌方式选择算子依据个体的适应度进行择优保留,并且是以概率方式选择个体,而不是确定性的选择。这使得算法一方面保证了群体向更优的方向进化,向最优解逼近,从而实现群体的优化更新;另一方面保证了基因库的多样性,为搜索出更优秀的个体提供条件。

群体智能算法是一种概率搜索算法,与传统的优化方法有很大的不同,群体智能算法在进行问题求解时,其最大特点是不依赖于问题本身的严格数学性质,它不要求所研究的问题是连续的、可导的,不需要建立关于问题本身的精确数学描述模型,不依赖于知识表示,一般不需要关于命题的先验知识的启发,而是在信号或数据层直接对输入信息进行处理,属于求解那些难以有效建立形式化模型、使用传统方法难以解决或根本不能解决的问题。

与传统的优化方法相比,群体智能算法具有以下优势。

(1) 渐进式寻优。

群体智能算法从随机产生的初始可行解出发,一代一代地反复迭代计算,使新一代的结果优越于上一代,逐渐得出最优的结果,这是一个逐渐寻优的过程,但是却可以很快地找出所要求的最优解。

(2) 体现“适者生存,劣者消亡”的自然选择规律。

在搜索过程中,借助群体选择操作,或个体变化前后的比较操作,无需添加任何额外的作用,就能使群体的品质不断地得到改进,具有自动适应环境的能力。

(3) 有指导的随机搜索。

群体智能算法是一种随机概率型的搜索方法,这种随机搜索既不是盲目搜索,也不是穷举式的全面搜索,而是一种有指导的随机搜索,指导算法执行搜索的依据是适应度,也就是它的

目标函数。在适应度的驱动下,利用概率来指导它的搜索方向,概率被作为一种信息来引导搜索过程朝着更优化的区域移动,使算法逐步逼近目标值。虽然表面看起来群体智能算法是一种盲目搜索方法,但实际上有着明确的搜索方向,这种不确定性使其能有更多的机会求得全局最优解。群体智能算法充分利用个体局部信息和群体全局信息,具有协同搜索的特点,搜索能力强。

(4) 并行式搜索。

群体智能算法具有并行性,表现在两个方面:一是内在并行性,即搜索过程是从一个解集合开始的,每一代运算都是针对一组个体同时进行,不容易陷入局部最优解。使其本身适合大规模的运算,让多台计算机各自独立运行种群的进化运算,适合在目前所有的并行机或分布式系统上并行处理,且容易实现,提高了算法的搜索速度。二是内含并行性,各种群分别独立进化,不需要相互之间进行信息交换,可以同时搜索解空间的多个区域,并相互交流信息,使得算法能以较少的代价获得较大的收益。

(5) 黑箱式结构。

群体智能算法直接表达问题的解,只研究输入与输出的关系,结构简单,并不深究造成这种关系的原因,算法根据所解决问题的特性,用字符串表达问题及选择适应度,个体的字符串表达如同输入,适应度计算如同输出,一旦完成这两项工作,其余的操作都可按固定方式进行。因此,从某种意义上讲,群体智能算法是一种只考虑输入与输出关系的黑箱问题,因此便于处理因果关系不明确的问题。群体智能算法对初值、参数选择不敏感,鲁棒性较强。

(6) 全局最优解。

群体智能算法由于采用群体搜索的策略,多点并行搜索,扩大了解的搜索空间,而且每次迭代模仿生物进化或觅食方式产生多种操作算子,在操作算子的作用下产生新个体,不断扩大搜索范围,具有极好的全局搜索性能。群体具有记忆个体最优解的能力,将搜索重点集中于性能高的部分,能够以很大的概率找到问题最优解。同时,算法中仅使用了问题的目标函数,对搜索空间有一定的自适应能力。因此群体智能算法很容易搜索出全局最优解而不是局部最优解,具有较好的全局寻优能力,提高了解的质量。

(7) 通用性强

传统的优化算法,需要将所解决的问题用数学式子表示,而且要求该函数的一阶导数或二阶导数存在。采用群体智能算法,只用某种编码表达问题,然后根据适应度区分个体优劣。其余的操作都是统一的,由计算机自动执行。因此有人称群体智能算法是一种框架式算法,它只有一些简单原则要求,在实施过程中,无需额外的干预,算法具有较强的通用性,使其不过分依赖于问题的信息。

(8) 智能性。

确定进化方案之后,群体智能算法不需要事先描述问题的全部特征,利用得到的信息自行组织搜索,基于自然选择策略,优胜劣汰,具备根据环境的变化,自动发现环境的特征和规律的能力,可用来解决未知结构的复杂问题。也就是说,群体智能算法能适应不同的环境、不同的问题,并且在大多数情况下都能得到比较有效的解。群体智能算法提供了噪声忍耐、无教师学习、自组织等进化学习机理,能够明晰地表达所学习的知识和结构,具有一些优良特性,如分布

式、并行性、自学习、自适应、自组织、鲁棒性和突显性等。除此之外,群体智能算法的优点还包括过程性、不确定性、非定向性、整体优化、稳健性等多个方面。群体智能算法在寻优等方面有着收敛速度快、鲁棒性好、全局收敛、适应范围宽等特点,可以适用于多种类型的优化问题。

(9) 具有较强的鲁棒性。

群体智能算法具有极强的容错能力,算法的初始种群可能包含与最优解相差很远的个体,但算法能通过选择策略,剔除适应度很差的个体,使可行解不断向最优解逼近。个体之间通过非直接的交流方式来进行合作,确保了系统具有更好的可扩展性和安全性;整个问题的解不会因为个体的故障受到影响,没有集中控制的约束,使得系统具有较强的鲁棒性。

(10) 易于与其他算法相结合。

较其他优化算法,群体智能算法控制参数少,原理相对简单,完全采用分布式来控制个体与个体、个体与环境之间的信息交互,具有良好的自组织性。由于系统中单个个体的能力十分简单,只需要最小智能,这样每个个体的执行时间较短。算法对问题定义的连续性无特殊要求,实现简单,易于与其他智能计算方法相结合,可以方便地将其他方法特有的一些操作算子直接并于其中,当然也可以很方便地与其他各种算法相结合产生新的优化算法。

1.6 模式识别的应用

我们在生活中时时刻刻都在进行模式识别,如识物、辨声、辨味等行为均属于模式识别的范畴。计算机出现后,人们企图用计算机来实现人或动物所具备的模式识别能力。当前主要是模拟人的视觉能力、听觉能力和嗅觉能力,如现在研究比较热门的图像识别技术和语音识别技术。这些技术已被广泛应用于军事与民用工业中。模式识别已经广泛应用于文字识别、语音识别、指纹识别、遥感、医学诊断、工业产品检测、天气预报、卫星航空图片解释等领域,近年来,用模式识别方法发展起来的“模式识别优化技术”在化工、冶金、石化、轻工等领域用于配方、工艺过程的优化设计和优化控制,产生了巨大的经济效益。在节约原料、提高产品质量和产量、降低单位能耗等方面充分显示了这一高新技术的巨大潜力。模式识别技术除了可以对配方、工艺进行优化设计外,还可以用于工业过程控制,这就是模式识别智能控制优化专家系统。它的特别是根据目标(例如降低能耗、提高产量等),优化影响目标的参量(如原料的组成、工艺参数等),在众多影响参量中筛选出对目标具有较重要影响的参量。经过模式分类、网络训练,确定优化区域,找出优化方向,动态建立模型,定量预报结果,使生产操作条件始终保持在优化状态,尽可能地挖掘生产潜力,在过程工业(包括化工、冶金、轻工、建材等)有广阔的应用前景。

所有这些应用都是和问题的性质密切不可分的,至今还没有发展成统一的、有效的可应用于所有的模式识别的理论。当前的一种普遍看法是不存在对所有的模式识别问题都使用的单一模型和解决识别问题的单一技术,我们现在拥有的是一个工具袋,我们所要做的是结合具体问题把模式识别方法结合起来,把模式识别与人工智能中的启发式搜索结合起来,把人工神经网络、不确定方法、智能计算结合起来,深入掌握各种工具的效能和应用的可能性,互相取长补短,开创模式识别应用的新局面。

模式识别技术是人工智能的基础技术,21世纪是智能化、信息化、计算化、网络化的世纪,在这个以数字计算为特征的世纪里,作为人工智能技术基础学科的模式识别技术,必将获得巨

大的发展空间。

本章小结

假定有一批待识别的事物,事先也不知道相关的先验知识,即不知道它们属于何种类别,满足何种分布,在这种情况下我们对这批事物分类的方法就是按照它们特征之间的相似性,将有相同或相似特征的事物聚集在一起,也就是说最后的分类结果中每一类聚集的物体都有共同的特征,这种不知所属类别而对根据事物相似性的程度分类的方法称为聚类。例如手写了15个数字(0,2,3,0,0,2,3,2,2,0,3,3,3,2,0),通过模式识别会把它们归成(0,2,3)3个类,这种方法叫做非监督学习方法。如果给定了一批待识别的事物,而且还知道了某些事物的类别,根据已知事物特征及其类别判断未知事物的类别,这种问题称为分类问题,分类与聚类的不同点是类数是确定的,而且已经知道了一批已经分类的事物。例如数字有固定的类数(0~9),能够识别出手写数字为哪一类,这种方法叫做监督学习方法。

监督学习方法用来对数据实现分类,分类规则通过训练获得。该训练集由带分类号的数据集组成,因此监督学习方法的训练过程是离线的。非监督学习方法不需要单独的离线训练过程,也没有带分类号的训练数据集,一般用来对数据集进行聚类分析,确定其分布。

总之,分类与聚类的效果好坏,一般说来最基本的性能评估是其错误率,如果能用反映错误率大小的准则,在理论上是最合适的。但是正如在前面讨论中提到的,对错误率的计算是极其复杂的,以至于很难构建直接基于错误率的判据。而且分类与聚类效果还受所使用的训练样品集,以及所用的算法影响,通常需要靠实践来检验。

本章介绍了设计分类器需要考虑的基本问题,包括特征空间优化设计问题、分类器设计准则、分类器设计基本方法、判别函数、分类器的选择和训练与学习,还介绍了聚类判别所涉及到的基本问题。这些都是模式识别需要考虑的重要内容,掌握这些内容为理解及实践后续各章所介绍的理论打下基础。

习题1

1. 简述特征空间优化的方法。
2. 简述几种常用的分类器设计准则。
3. 简述分类器设计的基本方法。
4. 试写出基于二维特征两类分类问题的线性判别函数形式。
5. 试写出基于 n 维特征两类分类问题的线性判别函数形式。
6. 试写出基于 n 维特征多类分类问题的线性判别函数形式。
7. 试写出基于 n 维特征多类分类问题的非线性判别函数形式。
8. 简述设计判别函数需要确定的基本要素。
9. 简述在什么情况下分类器不可分。
10. 简述设计一个分类器的基本方法。

第 2 章 特征的选择与优化

本章要点:

- ☑ 特征空间优化设计问题
- ☑ 样本特征库初步分析
- ☑ 样品筛选处理
- ☑ 特征筛选处理
- ☑ 特征评估
- ☑ 基于主成分分析的特征提取
- ☑ 特征空间描述与分析
- ☑ 手写数字特征提取与分析

在实际的应用中,信息采集的对象多数是多特征、高噪声、非线性的数据集。人们只能尽量多列一些可能有影响的因素,在样本数不是很多的情况下,用很多特征进行分类器设计,无论从计算的复杂程度还是就分类器性能来看都是不适宜的。因此,研究如何把高维特征空间压缩到低维特征空间就成为了一个重要的课题。任何识别过程的第一步,不论用计算机还是由人去识别,都要首先分析各种特征的有效性并选出最具有代表性的特征。

特征的选择与优化是非常重要的,它强烈地影响到分类器的设计及其性能。若对于不同类别样品特征的差别很大,那就比较容易设计出具有较高性能的分类器。因此,特征的选择是模式识别中的一个关键问题。由于在很多实际问题中常常不容易找到那些最重要的特征,或受条件限制不能对它们进行测量,这就使特征选择和优化的任务复杂化而成为构造模式识别系统最困难的任务之一。这个问题已经越来越受到人们的重视。

2.1 特征空间优化设计问题

特征选择和优化的基本任务是如何从许多特征中找出那些最有效的特征。解决特征选择和特征优化问题,最核心的内容就是如何对现有特征进行评估,以及如何通过现有特征产生更好的特征。在实际应用中,对特征选择与处理大致过程如图 2-1 所示。

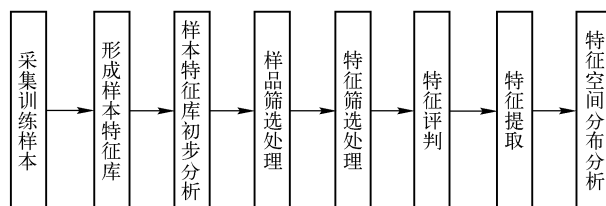


图 2-1 特征选择与处理过程

特征选择与优化过程如下:

(1) 对样本特征库进行初步分析是指对原始数据进行抽取,抽取那些对区别不同类别最为重要的特征,而舍去那些对分类并无多大贡献的特征,从而得到能反映分类本质的特征。考查所选特征是否合理,能否实现分类。如果把区别不同类别的特征都从输入数据中找到,这时自动模式识别问题就简化为匹配和查表,模式识别就不困难了。

(2) 对样本筛选处理的目的是去掉“离群点”,减少这些“离群点”对分类器的干扰。当条件所限无法采集大量的训练样品时,应慎重对待离群点。样本在特征空间中的理想分布是同类相聚、异类远离,但是在现实中很难达到理想的分布状态,就要求分类器具有泛化。

(3) 特征筛选处理的目的是分析特征之间的相关性,考查每个特征因子与目标有无关系,以及特征因子之间是否存在相关关系。删去那些相关的因子,在样本不多的条件下可以改善分类器的总体性能,降低模式识别系统的代价。特征的选择常常面临着保留哪些描述量、删除哪些描述量,通常要经过从多到少的过程。因为在设计识别方案的初期阶段,应该尽量多的列举出各种可能与分类有关的特征,这样可以充分利用各种有用的信息,改善分类效果。但大量的特征中肯定会包含许多彼此相关的因素,造成特征的重复和浪费,给计算带来困难。Kanal. L 曾经总结过经验:样品数 N 与特征数 n 之比应足够大,通常样本数 N 是特征数 n 的 5~10 倍。

(4) 特征评判的目的是分析经过筛选之后的特征,对分类效果能否提高,能否拉大不同类别之间的距离。对一个模式类特征选择的好与坏,很难在事先完全预测,而只能从整个分类识别系统获得的分类结果给予评价。

(5) 特征提取目的是用较少的特征对样本进行描述,以达到降低特征空间维数的目的。

(6) 接着需要进一步掌握样本库的总体分布情况,若发现效果不理想,应再一次考察样本库,或重新提取特征,或增加特征,或进一步删除“离群点”等。

如何确定合适的特征空间是设计模式识别系统十分重要、甚至更为关键的问题。如果所选用的特征空间能使同类物体分布具有紧致性,即各类样本能分布在该特征空间中彼此分割开的区域内,这就为分类器设计成功提供了良好的基础。反之,如果不同类别的样本在该特征空间中混杂在一起,再好的设计方法也无法提高分类器的准确性。

2.2 样本特征库初步分析

在模式识别处理之前,需要先评估一下特征库是否包含足够信息,用它做模式识别是否可行或值得。

1. 对样本数量与特征数目要求

通常要求样本数量 N 要足够大,符合下列关系。

① 对两类分类问题: $\frac{N}{n} \geq 3$,此处 n 为特征数目, N 为样本的数量。

② 对线性或非线性回归问题: $N \gg n$ 。

若实际课题中,由于不能确定哪些因素有影响,只能选择过多的特征,以致样本数量 N 不合乎上述要求。在无法获得足够多的样本情况下,应考虑下列两个措施。

① 通过特征筛选去除一批对目标影响小的特征,使 n 减少。

② 通过原理方面的论证或试探性地将若干特征组合成数目较少的特征。

2. 对样本特征库做初步分析

对样本特征库做初步分析的主要工作是衡量各类别之间的可分性,最常用的方法是应用“KNN 留一法”判据做近邻分析。KNN 留一法是以每个样品点与其多数最近邻属于同类与否作为判据。

根据样品在多维空间中的位置,计算各样品之间的距离,找出样品的三个、五个或多个最近邻,列表显示该样品的类别及近邻的类别,判断该最近邻是否属于同类,将多个同类的样品所属的类别作为预报该样品的类别,并与实际类别比较,仔细考查近邻分析结果,可对数据结构有一个大致的了解。如果样本在特征空间中分散,则需要选择具有泛化能力强的分类器,如神经网络分类器、支持向量机分类器等。

2.3 样品筛选处理

通常将“离群点”称为噪声,噪声干扰可能带来严重的后果。例如,使拟合度最佳的标准导致失误,或使真正有效的数学模型反而比“假”模型拟合度差些。用预报结果检验可能会甄别此事。统计学上,样品筛选处理的目的是删去某些离群的样品点,改善分类效果。定义和判断“离群点”的方法有以下几种。

① 若样本特征呈近线性关系,可用稳健回归方法确定“离群点”。

② 若样本特征不呈近线性关系,通常将近邻多半为异类的样品删除,或将其目标值与各近邻平均值相差特别大的样品删除,也可以将特征压缩后做回归分析。

上述方法由于认定和删去离群点基于若干假定,事先无法确定这些假定是否合乎实际,因此对删除后的数据必须谨慎对待。在实践中若能对离群点是否为“真”离群点做反复验证,才能增加结果的可靠性。经过初步评估,对“可分性”不满意时,可试行“样品筛选”操作,改善可分性。

2.4 特征筛选处理

在实际应用中,人们只能尽量多列一些可能有影响的因素,然后通过数据处理,考查和筛选出作用较大的特征,删去影响不大的特征,从而建立数学模型。特征筛选的第一步是对每个特征做分析,考查每个特征与目标的相关性,特征与特征之间的相关性。

用原始变量为坐标作投影图,考查单个特征、双特征、多特征对目标值的影响,计算相关系数。

1. 单特征相关分析

将所有特征逐个对目标值作二维图,计算目标值 t 与特征 x_j 之间的相关系数

$$r(t, x_j) = \frac{\sum_{i=1}^N (t_i - \bar{t})(x_{ij} - \bar{x}_j)}{\left[\sum_{i=1}^N (t_i - \bar{t})^2 \sum_{i=1}^N (x_{ij} - \bar{x}_j)^2 \right]^{\frac{1}{2}}} \quad (2-1)$$

式中, i 为样品号数; t_i, x_{ij} 为第 i 个样品的目标值和第 j 个特征值; \bar{t} 和 \bar{x}_j 分别为所有样本的目标值的平均值和第 j 个特征的平均值; 相关系数 $r(t, x_j)$ 介于 1 与 -1 之间, 作为最简单的近似方法, 各特征的重要性可用相关系数的绝对值大小评估。

根据特征对目标值或分类的影响大小, 删去作用小、噪声大的变量。为了不漏掉重要因子, 一开始我们宁可多选一些特征, 然后根据各个特征在描述研究对象时作用的大小, 删去那些带来信息少、噪声多的特征; 并将删后保留的特征按其描述对象关系的大小做一个大致的排序, 突出主要因素, 这对建立模式识别系统是十分必要的。特征筛选的原理是: 一个原有 $n+1$ 个特征的特征库, 删去其中一个特征, 得到一个特征数为 n 的新数据库; 若删去的变量贡献的信息小于带来的噪声量, 删去后信息量未显著减少或反而增加, 则该特征为可删变量。

2. 双特征相关分析

在所有特征中每次取出两个特征作为纵、横坐标作图, 同时将样本分为两类(或多类), 以不同符号显示于图中, 据此考查两类或多类样本在图中分布的规律; 同时还显示两个特征间的相关系数。

3. 三特征相关分析

在所有特征中每次选用三个, 作为 x, y, z 坐标作三维图, 同时将样本分为两类或多类, 以不同符号显示于图中, 据此考查各类样本在三维空间的分布规律; 也可选两个特征为 x 和 y 坐标, 目标值为 z 坐标, 考查其关系。三维结构可通过图形旋转考查, 同时显示旋转后的二维坐标与原始变量的关系。

4. 子空间局部考查

将原始多维空间“切割”为几个子空间, 然后再做相关分析, 往往能揭示重要的规律性。

因复杂系统往往是多特征问题, 目标值或目标类别往往由三个以上的因子共同决定, 单考查一个、两个或三个因子的影响往往不够, 因为由于其他因子(特征)变化的干扰, 往往不能有效地全面显示特征空间的规律性, 只有运用多种模式识别方法建模才能全面解决问题。但是作为初步考查手段, 相关分析方法(特别是与子空间局部考查结合后)很有用, 因为相关分析及其作图方法显示的是原始特征, 若能找到规律, 其物理(或化学)意义的诠释比较简单明了。各种模式识别方法虽能提供更完整可靠的数学模型, 但因其坐标表达式多为多个原始变量的线性或非线性组合, 诠释比较复杂。

相关分析的局限性: 根据前述的单特征相关分析方法, 删除相关系数小的特征。这种方法对于样本分布不均匀的特征库是不可靠的做法。如果目标与特征之间呈线性关系, 对于样本分布不均匀的数据文件, 单比较相关系数也不是绝对可靠的做法, 因为它没有考虑其他特征的影响。总而言之, 可以肯定的是: 若 x_i 与 t (或 x_j) 相关系数很大(如 0.5 以上或 -0.5 以下), x_i 肯定对 t (或 x_j) 有较大影响; 若相关系数较小, 则要参照其他信息才能决定是否可删。

5. 特征选择及搜索算法

特征选择的任务是从一组数量为 D 的特征中选择出数量为 n ($D > n$) 的一组最优特征来,一方面需要确定可分离性判据 $J(x)$,对特征选择效果做评估,选出使某一可分性达最大的特征组来(详见 2.4 节)。另一方面是要找到一个较好的算法,以便在允许的时间内找出最优的那一组特征。

如果采用穷举法,把 D 个特征每个单独使用时的可分性判据都算出来,按判据大小排队,例如

$$J(x_1) > J(x_2) > \cdots > J(x_n) > \cdots > J(x_D)$$

单独使用时使 J 较大的前 n 个特征作为特征组并不具有最优的效果,甚至有可能是最不好的特征组。

从 D 个特征中挑选 n 个,所有可能的组合数为

$$q = C_D^n = \frac{D!}{(D-n)!n!} \quad (2-2)$$

如果把各种可能的特征组合的 J 都算出来再加以比较,以选择最优特征组,则计算量太大而无法实现。这就使得寻找一种可行的算法变得非常必要。

应当说明的是,任何非穷举的算法都不能保证所得结果是最优的。因此,除非只要求次优解,否则所选算法原则上仍是穷举算法,只不过采取某些搜索技术使计算量可能有所降低。在所有算法中,最优特征组的构成都是用每次从现存特征中增加或去掉某些特征的方法直至特征数等于 n 为止,若特征数从零逐步增加则称为“自下而上”法。反之,若从特征数 D 开始逐步减少,则称为“自上而下”法。

令 Φ_k 表示特征数目为 k 的所有可能的特征组合, $\bar{\Phi}_k$ 表示从 x_1, x_2, \cdots, x_D 去掉 k 个后所剩特征的所有可能的特征组合。

在“自下而上”算法中第 k 步的最优特征组应当使

$$J(\Phi'_k) = \max_{|\Phi_k|} J(\Phi_k)$$

从 $\Phi_0 = \emptyset$ 开始, $k=1, 2, \cdots$, 直到 $k=n$, 结果得

$$\Phi = \Phi_n$$

在“自上而下”算法中第 k 步的最优特征组应当使

$$J(\bar{\Phi}'_k) = \max_{|\bar{\Phi}_k|} J(\bar{\Phi}_k)$$

从 $\bar{\Phi} = \Phi_D$ 开始, $k=1, 2, \cdots$, 直到 $k=D-n$, 结果所得特征组为 $\Phi = \bar{\Phi}_{D-n}$ 。

2.5 特征评估

对原特征空间进行优化之后,就要对优化的结果进行评价,通过反复选择不同的特征组合,采用定量分析比较的方法,判断所得到的特征维数,及所使用特征是否对分类最有利,这种用以定量检验分类性能的准则称为类别可分离性判据,用来检验不同的特征组合对分类性能好坏的影响。对特征空间进行优化是一种计算过程,它的基本方法仍然是模式识别的典型方法,即找到一种准则(或称判据),通常用一种式子表示,使这种计算准则达到一个极值。对特征评估的方法大体分两类:一类以计算样品在特征空间离散程度为基础的准则,称为基于距离

的可分性判据;另一类则基于概率密度分布的判据。

下面介绍基于距离的可分性判据。

给定一组表示联合分布的训练集,假定每一类的模式向量在观察空间中占据不同的区域是合理的,类别模式间的距离或平均距离则是模式空间中类别可分离性的度量。基于距离的可分性判据的出发点:各类样本之间的距离越大、类内散度越小,则类别的可分性越好。基于距离的可分性判据直接依靠样本计算,直观简捷,物理概念清晰,因此目前应用较为广泛。

在一个特征候选集 $\mathbf{X} = [x_1, x_2, \dots, x_n]$ 所定义的 n 维特征空间中,用 $d(\mathbf{X}_{ik}, \mathbf{X}_{jl})$ 表示第 i 类中第 k 个样品和第 j 类中第 l 个样品间距离的度量值,距离度量 d 可采用式(2-3)定义的欧几里德距离计算:

$$d(\mathbf{X}_{ik}, \mathbf{X}_{jl}) = \left[\sum_{m=1}^D (x_{ik,m} - x_{jl,m})^2 \right]^{1/2} (i, j = 1, 2, \dots, M; k = 1, 2, \dots, N_i; l = 1, 2, \dots, N_j) \quad (2-3)$$

类间的平均距离可采用式(2-4)计算:

$$J = \frac{1}{2} \sum_{i=1}^M \sum_{j=1}^M P(\omega_i) P(\omega_j) \cdot \frac{1}{N_i N_j} \sum_{k=1}^{N_i} \sum_{l=1}^{N_j} d(\mathbf{X}_{ik}, \mathbf{X}_{jl}) \quad (2-4)$$

考虑到式(2-4)的计算比较复杂,可将其转化为相应的矩阵来度量和处理。

(1) 总体散布矩阵

① 第 i 类均值向量

$$\overline{\mathbf{X}^{(\omega_i)}} = \frac{1}{N_i} \sum_{\mathbf{X} \in \omega_i} \mathbf{X} \quad (2-5)$$

② 样本集总体均值向量

$$\bar{\mathbf{X}} = \frac{1}{N} \sum_{i=1}^N \mathbf{X}_i = \frac{1}{N} \sum_{i=1}^M P(\omega_i) \overline{\mathbf{X}^{(\omega_i)}} \quad (2-6)$$

③ 第 i 类协方差

$$\sum_i = \frac{1}{N_i - 1} \sum_{\mathbf{X} \in \omega_i} (\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}}) (\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}})^T \quad (2-7)$$

④ 样本总体协方差

$$\sum = \frac{1}{N - 1} \sum (\mathbf{X} - \bar{\mathbf{X}}) (\mathbf{X} - \bar{\mathbf{X}})^T \quad (2-8)$$

⑤ 第 i 类类内散布矩阵

$$\mathbf{S}_i = E \left\{ (\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}}) (\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}})^T \right\} = \sum_i \quad (2-9)$$

⑥ 总体类内散布矩阵

$$\begin{aligned} \mathbf{S}_W &= \sum_{i=1}^M P(\omega_i) \mathbf{S}_i = \sum_{i=1}^M P(\omega_i) E \left\{ (\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}}) (\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}})^T \right\} \\ &= \sum_{i=1}^M P(\omega_i) \sum_i \end{aligned} \quad (2-10)$$

⑦ 总体类间散布矩阵

$$\mathbf{S}_B = \sum_{i=1}^M P(\omega_i) (\overline{\mathbf{X}^{(\omega_i)}} - \bar{\mathbf{X}})(\overline{\mathbf{X}^{(\omega_i)}} - \bar{\mathbf{X}})^T \quad (2-11)$$

特别对于两类问题

$$\mathbf{S}_{B2} = (\overline{\mathbf{X}^{(\omega_1)}} - \overline{\mathbf{X}^{(\omega_2)}})(\overline{\mathbf{X}^{(\omega_1)}} - \overline{\mathbf{X}^{(\omega_2)}})^T \quad (2-12)$$

⑧ 总体散布矩阵

$$\mathbf{S}_T = E\{(\mathbf{X} - \bar{\mathbf{X}})(\mathbf{X} - \bar{\mathbf{X}})^T\} = \sum \quad (2-13)$$

存在关系

$$\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B \quad (2-14)$$

类内散布矩阵表征各样本点围绕它的均值的散布情况,类间散布均值表征各类间的距离分布情况,它们依赖于样本类别属性和划分;而总体散布矩阵与样本划分及类别属性无关。

(2) 构造准则

以类内散布矩阵 \mathbf{S}_W , 类间散布矩阵 \mathbf{S}_B 和总体散布矩阵 \mathbf{S}_T 为基础的一些准则如下。

① 均方误差最小准则, 即迹准则

$$J = \text{tr} \mathbf{S}_W = \sum_{i=1}^M P(\omega_i) \text{tr} \mathbf{S}_i \quad (2-15)$$

或

$$J = \det(\mathbf{S}_W) \quad (2-16)$$

② 类间距离最大准则

$$J = \text{tr}(\mathbf{S}_B) \text{ 或 } J = \det(\mathbf{S}_B) \quad (2-17)$$

③ 行列式准则

$$J = |\mathbf{S}_W| = \sum_{i=1}^M P(\omega_i) |\mathbf{S}_i| \quad (2-18)$$

基于距离的可分性判据的出发点是: 各类样本之间的距离越大、类内散度越小, 则类别的可分性越好。基于距离的可分性判据直接依靠样本计算, 直观简捷, 物理概念清晰, 因此目前应用较为广泛。

2.6 基于主成分分析的特征提取

在模式识别问题中, 对于初始特征的选择, 绝大多数都是在考虑样本的可分性意义上进行的。所以很多时候选择的初始特征集合都会包含大量互相关联的特征, 它们对于样本分类的贡献也是很不相同的。大的特征向量集合有很多的不便, 最明显的就是计算方面会有很大负担。所以, 在模式识别问题中, 通常的任务就是进行特征的选择。在最初的模式识别工程中, 这种选择有两个目标: 或者丢弃一些对分类贡献不大的特征; 或者达到一定程度降维的目的, 降维的方法通常是采用一个从初始特征衍生得到的、更小的、与原特征集相当的特征集合。

主成分分析是把多个特征映射为少数几个综合特征的一种统计分析方法。在多特征的研究中, 往往由于特征个数太多, 且彼此之间存在着一定的相关性, 因而使得所观测的数据在一定程度上有信息的重叠。当特征较多时, 在高维空间中研究样本的分布规律就更麻烦。主成分分析采取一种降维的方法, 找出几个综合因子来代表原来众多的特征, 使这些综合因子尽可

能地反映原来变量的信息,而且彼此之间互不相关,从而达到简化的目的。

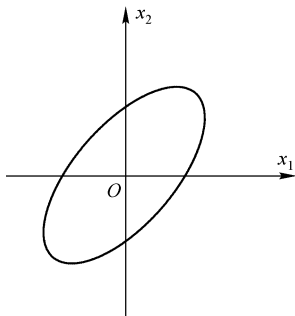


图 2-2 二维空间主成分示意图

1. 主分量的几何解释

如果从研究总体中抽取 N 个样品,每个样品有 2 个指标,这时的主分量实际上是二维空间的几何意义问题。

设 N 个样品在二维空间中的分布大致为一个椭圆,如图 2-2 所示。将坐标系正交旋转一个角度 θ ,在椭圆长轴方向取坐标 y_1 ,在短轴方向取坐标 y_2 ,则旋转公式为

$$y_{1j} = x_{1j}\cos\theta + x_{2j}\sin\theta \quad (2-19)$$

$$y_{2j} = x_{1j}(-\sin\theta) + x_{2j}\cos\theta \quad (2-20)$$

式中, $j=1,2,\dots,N$ 。写成矩阵形式为

$$Y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1N} \\ y_{21} & y_{22} & \cdots & y_{2N} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \end{bmatrix} = UX$$

其中, U 为坐标旋转变换矩阵,它是正交变换矩阵,即有 $U^T = U^{-1}$, $UU^T = I$ 。经过旋转变换后得到如图 2-3 所示的新坐标。从图 2-3 可以看出:

- ① N 个点的坐标 y_1 和 y_2 的相关几乎为零;
- ② 二维平面上 N 个点的方差大部分都归结在 y_1 轴上,而 y_2 轴上的方差较小。

y_1 和 y_2 是原始变量 x_1 和 x_2 的综合变量。

由于 N 个点在 y_1 轴上的方差最大,因而用在 y_1 轴上的一维综合变量来代替,二维空间的点所损失的信息量最小,由此称 y_1 轴为第一主分量, y_2 轴与 y_1 轴正交且有较小的方差,称它为第二主分量。

一般说来,如果 N 个样品中的每个样品有 n 个特征 x_1, x_2, \dots, x_n , 经过主成分分析,将它们综合成 n 个综合变量,即

$$\begin{cases} y_1 = c_{11}x_1 + c_{12}x_2 + \cdots + c_{1n}x_n \\ y_2 = c_{21}x_1 + c_{22}x_2 + \cdots + c_{2n}x_n \\ \vdots \\ y_n = c_{n1}x_1 + c_{n2}x_2 + \cdots + c_{nn}x_n \end{cases} \quad (2-21)$$

并且满足 $c_{k1}^2 + c_{k2}^2 + \cdots + c_{kn}^2 = 1$ ($k=1,2,\dots,n$), 其中 c_{ij} 由下列原则决定:

- ① y_i 与 y_j ($i \neq j; i, j=1,2,\dots,n$) 相互独立。
- ② y_1 是 x_1, x_2, \dots, x_n 满足式 (2-21) 的一切线性组合中方差最大者, y_2 是与 y_1 不相关的 x_1, x_2, \dots, x_n 的所有线性组合中方差次大者,以此类推, y_n 是与 y_1, y_2, \dots, y_{n-1} 都不相关的 x_1, x_2, \dots, x_n 的所有线性组合中方差最小者。

这样决定的综合指标因子 y_1, y_2, \dots, y_n 分别被称为原变量的第 1、第 2、 \dots 、第 n 个主分量, 它们的方差依次递减。

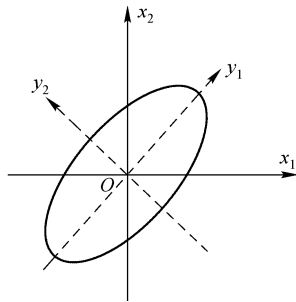


图 2-3 二维空间主成分正交示意图

2. 主分量的导出

设 $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ 是一个 n 维随机向量, $\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$ 是满足式(2-21)的新变量所构成的向量。于

是式(2-21)的矩阵形式为 $\mathbf{Y} = \mathbf{C}\mathbf{X}$, \mathbf{C} 为正交矩阵, 并满足 $\mathbf{C}\mathbf{C}^T = \mathbf{I}$, \mathbf{I} 为单位矩阵。

坐标旋转是指新坐标轴相互正交, 仍构成一个直角坐标系。变换后的 N 个点在 y_1 轴上有最大方差, 在 y_2 轴上有次大方差, 以此类推, 在 y_n 轴上有最小的方差。同时, N 个点对不同的 y_i 轴和 y_j 轴的协方差 ($j \neq i$) 为零, 即要求 \mathbf{Y} 的协方差

$$\mathbf{Y}\mathbf{Y}^T = (\mathbf{C}\mathbf{X})(\mathbf{C}\mathbf{X})^T = \mathbf{C}\mathbf{X}\mathbf{X}^T\mathbf{C}^T = \mathbf{A} \quad (2-22)$$

其中

$$\mathbf{A} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}$$

假定 \mathbf{X} 为已标准化处理后的数据矩阵, 则 $\mathbf{X}\mathbf{X}^T$ 为原始数据的相关矩阵。令 $\mathbf{R} = \mathbf{X}\mathbf{X}^T$, 则式(2-22) 表示为 $\mathbf{C}\mathbf{R}\mathbf{C}^T = \mathbf{A}$ 。由 \mathbf{C}^T 左乘该式, 有

$$\mathbf{R}\mathbf{C}^T = \mathbf{C}^T\mathbf{A} \quad (2-23)$$

写成代数式为

$$\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nn} \end{bmatrix} \times \begin{bmatrix} c_{11} & c_{21} & \cdots & c_{n1} \\ c_{12} & c_{22} & \cdots & c_{n2} \\ \vdots & \vdots & & \vdots \\ c_{1n} & c_{2n} & \cdots & c_{nn} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{21} & \cdots & c_{n1} \\ c_{12} & c_{22} & \cdots & c_{n2} \\ \vdots & \vdots & & \vdots \\ c_{1n} & c_{2n} & \cdots & c_{nn} \end{bmatrix} \times \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

将上式全部展开得到 n^2 个方程, 这里考虑在矩阵乘积中由第 1 列得出的 n 个方程为

$$(r_{11} - \lambda_1)c_{11} + r_{12}c_{12} + \cdots + r_{1n}c_{1n} = 0$$

$$r_{21}c_{11} + (r_{22} - \lambda_1)c_{12} + \cdots + r_{2n}c_{1n} = 0$$

$$\vdots$$

$$r_{n1}c_{11} + r_{n2}c_{12} + \cdots + (r_{nn} - \lambda_1)c_{1n} = 0$$

为得到齐次方程组的非零解, 要求关于 c_{ij} 的系数行列式为 0, 即

$$\begin{vmatrix} r_{11} - \lambda_1 & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} - \lambda_1 & \cdots & r_{2n} \\ \vdots & \vdots & & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nn} - \lambda_1 \end{vmatrix} = 0$$

写成矩阵形式为 $|\mathbf{R} - \lambda\mathbf{I}| = 0$ 。对于 $\lambda_2, \lambda_3, \cdots, \lambda_n$, 可以得到完全类似的方程, 故 λ_j ($j=1, 2, \cdots, n$) 是 $|\mathbf{R} - \lambda\mathbf{I}| = 0$ 的 n 个根, λ 为特征方程的特征根, 相应的各个 c_{ij} 为其特征向量的分量。

设 \mathbf{R} 的 n 个特征值 $\lambda_1 > \lambda_2 > \cdots > \lambda_n \geq 0$, 相应于 λ_i 的特征向量为 \mathbf{C}_i , 令

$$\mathbf{C} = \begin{bmatrix} c_{11} & c_{21} & \cdots & c_{n1} \\ c_{12} & c_{22} & \cdots & c_{n2} \\ \vdots & \vdots & & \vdots \\ c_{1n} & c_{2n} & \cdots & c_{nn} \end{bmatrix} = [\mathbf{C}_1, \mathbf{C}_2, \cdots, \mathbf{C}_n]$$

相对于 y_1 的方差为

$$\text{Var}(\mathbf{C}_1 \mathbf{X}) = \mathbf{C}_1 \mathbf{X} \mathbf{X}^T \mathbf{C}_1^T = \mathbf{C}_1 \mathbf{R} \mathbf{C}_1^T = \lambda_1$$

同样有

$$\text{Var}(\mathbf{C}_i \mathbf{X}) = \lambda_i$$

即对于 y_1 有最大的方差, y_2 有次大的方差等, 并且有协方差

$$\text{Cov}(\mathbf{C}_i^T \mathbf{X}^T, \mathbf{C}_j \mathbf{X}) = \mathbf{C}_i^T \mathbf{R} \mathbf{C}_j \quad (2-24)$$

由式(2-23)得 $\mathbf{R} = \sum_{a=1}^n \lambda_a \mathbf{C}_a \mathbf{C}_a^T$, 所以式(2-24)变为

$$\text{Cov}(\mathbf{C}_i^T \mathbf{X}^T, \mathbf{C}_j \mathbf{X}) = \mathbf{C}_i^T \mathbf{R} \mathbf{C}_j = \mathbf{C}_i^T \left(\sum_{a=1}^n \lambda_a \mathbf{C}_a \mathbf{C}_a^T \right) \mathbf{C}_j = \sum_{a=1}^n \lambda_a (\mathbf{C}_i^T \mathbf{C}_a) (\mathbf{C}_a^T \mathbf{C}_j) = 0 (i \neq j)$$

变量 x_1, x_2, \cdots, x_n 经过正交变换后得到新的随机向量

$$y_1 = \mathbf{C}_1^T \mathbf{X}$$

$$y_2 = \mathbf{C}_2^T \mathbf{X}$$

$$\vdots$$

$$y_n = \mathbf{C}_n^T \mathbf{X}$$

y_1, y_2, \cdots, y_n 彼此不相关, 并且 y_i 的方差为 λ_i , 故称 y_1, y_2, \cdots, y_n 分别为第 1、第 2、 \cdots 、第 n 个主分量。

第 i 个主分量的贡献率定义为 $\lambda_i / \sum_{k=1}^n \lambda_k (i = 1, 2, \cdots, n)$, 前 m 个主分量的累积贡献率定义为 $\sum_{i=1}^m \lambda_i / \sum_{k=1}^n \lambda_k$, 选取前 $m (m < n)$ 个主分量, 使其累积贡献率达到一定的要求 (如 80% ~ 90%), 以前 m 个主分量代替原始数据做分析, 这样便可达到降低原始数据维数的目的。

2.7 特征空间描述与分析

2.7.1 特征空间描述

对于样本的特征空间描述, 主要分析特征的集中位置、分散程度、数据的分布为正态还是偏态等。对于多维数据, 还要分析多维数据的各个分量之间的相关性等。

1. 一维特征

一维样品的特征空间描述主要有以下几种。设 N 个观测值为 x_1, x_2, \cdots, x_N , 其中 N 称为样本容量。

① 均值, 即是 x_1, x_2, \cdots, x_N 的平均数

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i \quad (2-25)$$

② 方差,描述数据取值分散性的一个度量,它是数据相对于均值的偏差平方的平均值

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2 \quad (2-26)$$

③ 标准差,方差的开方称为标准差

$$s = \sqrt{s^2} = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2} \quad (2-27)$$

偏度与峰度是刻画数据的偏态、尾重程度的度量,它们与数据的矩有关。数据的矩分为原点矩与中心矩。

④ k 阶原点矩为

$$v_k = \frac{1}{N} \sum_{i=1}^N x_i^k \quad (2-28)$$

⑤ k 阶中心矩为

$$u_k = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^k \quad (2-29)$$

⑥ 偏度的计算公式为

$$g_1 = \frac{N}{(N-1)(N-2)s^3} \sum_{i=1}^N (x_i - \bar{x})^3 = \frac{N^2 u_3}{(N-1)(N-2)s^3} \quad (2-30)$$

其中 s 是标准差。偏度是刻画数据对称性的指标。关于均值对称的数据其偏度为0,右侧更分散的数据偏度为正,左侧更分散的数据偏度为负,如图2-4所示。

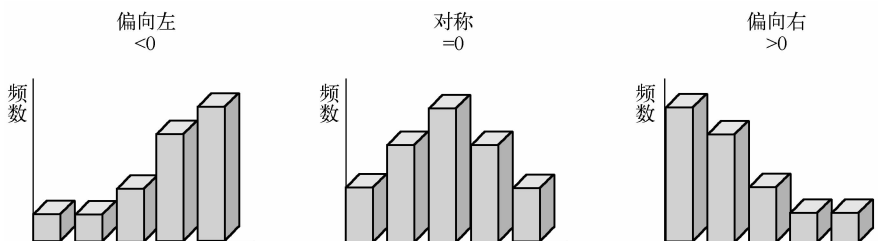


图2-4 偏度示意图

⑦ 峰度的计算公式为

$$\begin{aligned} g_2 &= \frac{N(N+1)}{(N-1)(N-2)(N-3)s^4} \sum_{i=1}^N (x_i - \bar{x})^4 - 3 \frac{(N-1)^2}{(N-2)(N-3)} \\ &= \frac{N^2(N+1)u_4}{(N-1)(N-2)(N-3)s^4} - 3 \frac{(N-1)^2}{(N-2)(N-3)} \end{aligned} \quad (2-31)$$

当数据的总体分布为正态分布时,峰度近似为0;当分布较正态分布的尾部更分散时,峰度为正,否则峰度为负。当峰度为正时,两侧极端数据较多;当峰度为负时,两侧极端数据较少。

2. 二维特征

设 $(\mathbf{X}, \mathbf{Y})^T$ 是二维总体,从中取得的观测数据 $(x_1, y_1)^T, (x_2, y_2)^T, \dots, (x_N, y_N)^T$ 。引进数

观测矩阵

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \\ y_1 & y_2 & \cdots & y_N \end{bmatrix} \quad (2-32)$$

① 二维观测数据的均值向量 $(\bar{x}, \bar{y})^T = \begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix}$

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i, \bar{y} = \frac{1}{N} \sum_{i=1}^N y_i \quad (2-33)$$

② 变量 \mathbf{X} 的观测数据的方差 s_{xx} , 变量 \mathbf{Y} 的观测数据的方差 s_{yy} 及变量 \mathbf{X}, \mathbf{Y} 的观测数据的协方差 s_{xy}

$$s_{xx} = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2, s_{yy} = \frac{1}{N-1} \sum_{i=1}^N (y_i - \bar{y})^2$$

$$s_{xy} = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y}) \quad (2-34)$$

③ 观测数据的协方差矩阵

$$\mathbf{S} = \begin{bmatrix} s_{xx} & s_{xy} \\ s_{yx} & s_{yy} \end{bmatrix} \quad (2-35)$$

注意: 总有 $s_{xy} = s_{yx}$, 即协方差矩阵为对称矩阵。

由 Schwarz 不等式

$$s_{xy}^2 \leq s_{xx} s_{yy}$$

所以 \mathbf{S} 总是非负定的, 一般是正定的。

④ 观测数据的相关系数:

$$r_{xy} = \frac{s_{xy}}{\sqrt{s_{xx}} \sqrt{s_{yy}}} \quad (2-36)$$

由 Schwarz 不等式, 有 $|r_{xy}| \leq 1$, 即总有 $-1 \leq r_{xy} \leq 1$ 。

例如, 有 10 名学生, 其中 5 名男生, 5 名女生。对每名学生取身高、体重两项指标作为特征, 测得的数据如表 2-1 所示。

表 2-1 学生数据

特征(指标) \ 样品(学生)	男 生					女 生				
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
x_1 (身高:m)	1.70	1.75	1.65	1.80	1.78	1.60	1.55	1.60	1.65	1.70
x_2 (体重:kg)	65	70	60	65	70	60	45	45	50	55

10 个样品的均值为

$$\bar{\mathbf{X}} = \frac{1}{10} \sum_{i=1}^{10} \mathbf{X}_i = (1.678, 58.5)^T$$

男生和女生样品点的均值为

$$\overline{\mathbf{X}^{(1)}} = \frac{1}{5} \sum_{i=1}^5 \mathbf{X}_i = (1.736, 66.0)^T$$

$$\overline{\mathbf{X}^{(2)}} = \frac{1}{5} \sum_{i=6}^{10} \mathbf{X}_i = (1.62, 51.0)^T$$

特征 x_1 对于全体样品的方差为

$$s_1^2 = \frac{1}{10-1} [(1.70 - 1.678)^2 + (1.75 - 1.678)^2 + \cdots + (1.70 - 1.678)^2] = 0.0068$$

特征 x_2 对于全体样品的方差为

$$s_2^2 = \frac{1}{10-1} [(65 - 58.5)^2 + (70 - 58.5)^2 + \cdots + (55 - 58.5)^2] = 89.1667$$

特征 x_1 对于男生和女生样品的方差为

$$s_1^2(1) = \frac{1}{5-1} [(1.70 - 1.736)^2 + (1.75 - 1.736)^2 + \cdots + (1.78 - 1.736)^2] = 0.0037$$

$$s_1^2(2) = \frac{1}{5-1} [(1.60 - 1.620)^2 + (1.55 - 1.62)^2 + \cdots + (1.70 - 1.62)^2] = 0.0032$$

特征 x_2 对于男生和女生样品的方差为

$$s_2^2(1) = \frac{1}{5-1} [(65 - 66)^2 + (70 - 66)^2 + \cdots + (70 - 66)^2] = 17.5$$

$$s_2^2(2) = \frac{1}{5-1} [(60 - 51)^2 + (45 - 51)^2 + \cdots + (55 - 51)^2] = 42.5$$

全体样品点中特征 x_1 与 x_1 的协方差 s_{11} 即 s_1^2 ; x_1 与 x_2 的协方差为

$$s_{12} = \frac{1}{10-1} [(1.70 - 1.678)(65 - 58.5) + (1.75 - 1.678)(70 - 58.5) + \cdots + (1.70 - 1.678)(55 - 58.5)] = 0.6356$$

全体样品点中特征 x_2 与 x_2 的协方差 s_{22} 即 s_2^2 ; 在男生和女生样品点中分别有

$$s_{12}(1) = \frac{1}{5-1} [(1.70 - 1.736)(65 - 66) + (1.75 - 1.736)(70 - 66) + \cdots + (1.78 - 1.736)(55 - 66)] = 0.18$$

$$s_{12}(2) = \frac{1}{5-1} [(1.60 - 1.620)(65 - 51.0) + (1.55 - 1.62)(45 - 51.0) + \cdots + (1.70 - 1.62)(55 - 51.0)] = 0.163$$

则对于全体样品点 x_1 和 x_2 的相关系数为

$$r_{12} = \frac{0.6356}{\sqrt{0.0068} \sqrt{89.1667}} = 0.8163$$

特征 x_1 和 x_2 对于男生和女生的相关系数为

$$r_{12}(1) = \frac{0.18}{\sqrt{0.0037} \sqrt{17.5}} = 0.7074$$

$$r_{12}(2) = \frac{0.1625}{\sqrt{0.0032} \sqrt{42.5}} = 0.4406$$

3. 多维特征

设 $(\mathbf{X}_{(1)}, \mathbf{X}_{(2)}, \cdots, \mathbf{X}_{(n)})$ 是 n 维总体, 从中取得样品数据

$$(x_{11}, x_{12}, \cdots, x_{1n})^T$$

$$\begin{pmatrix} x_{21}, x_{22}, \cdots, x_{2n} \\ \vdots \\ x_{N1}, x_{N2}, \cdots, x_{Nn} \end{pmatrix}^T$$

第 i 个观测数据记为

$$\mathbf{X}_i = (x_{i1}, x_{i2}, \cdots, x_{in})^T \quad i = 1, 2, \cdots, N$$

称为样品。引进样品数据观测矩阵

$$\tilde{\mathbf{X}} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Nn} \end{bmatrix} = [\mathbf{X}_{(1)}, \mathbf{X}_{(2)}, \cdots, \mathbf{X}_{(n)}]$$

它是 $N \times n$ 矩阵, 它的 N 个行即是 N 个样品 $\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_N$, 它们组成来自 n 维总体 $(\mathbf{X}_{(1)}, \mathbf{X}_{(2)}, \cdots, \mathbf{X}_{(n)})$ 的样品。观测矩阵 $\tilde{\mathbf{X}}$ 的 n 个列分别是 n 个变量 $\mathbf{X}_{(1)}, \mathbf{X}_{(2)}, \cdots, \mathbf{X}_{(n)}$ 在 N 次试验中所取的值。记为

$$\mathbf{X}_{(j)} = (x_{1j}, x_{2j}, \cdots, x_{Nj})^T \quad j = 1, 2, \cdots, n \quad (2-37)$$

(1) 样品统计参数

定义 $\bar{\mathbf{X}} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)^T$ 是 n 维样本数据的均值向量。

① 第 j 行 $\mathbf{X}_{(j)}$ 的均值

$$\bar{x}_j = \frac{1}{N} \sum_{i=1}^N x_{ij} \quad j = 1, 2, \cdots, n \quad (2-38)$$

② 第 j 行 $\mathbf{X}_{(j)}$ 的方差

$$s_j^2 = \frac{1}{N-1} \sum_{i=1}^N (x_{ij} - \bar{x}_j)^2 \quad j = 1, 2, \cdots, n \quad (2-39)$$

③ $\mathbf{X}_{(j)}, \mathbf{X}_{(k)}$ 的协方差

$$s_{jk} = \frac{1}{N-1} \sum_{i=1}^N (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k) \quad j, k = 1, 2, \cdots, n \quad (2-40)$$

$\mathbf{X}_{(j)}$ 与自身的协方差即 $\mathbf{X}_{(j)}$ 的方差

$$s_j^2 = s_{jj} \quad j = 1, 2, \cdots, n$$

称

$$\mathbf{S} = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1n} \\ s_{12} & s_{22} & \cdots & s_{2n} \\ \vdots & \vdots & & \vdots \\ s_{n1} & s_{n2} & \cdots & s_{nn} \end{bmatrix} \quad (2-41)$$

是样品观测数据的协方差矩阵。有

$$\mathbf{S} = \frac{1}{N-1} \sum_{i=1}^N (\mathbf{X}_i - \bar{\mathbf{X}})(\mathbf{X}_i - \bar{\mathbf{X}})^T \quad (2-42)$$

均值向量 $\bar{\mathbf{X}}$ 与协方差矩阵 \mathbf{S} 是 n 维观测数据的重要数字特征。 $\bar{\mathbf{X}}$ 表示 n 维观测数据的集中位置, 而协方差矩阵 \mathbf{S} 的对角线元素分别是各个变量观测值的方差, 而非对角线元素是变量观测值之间的协方差。

④ $X_{(j)}, X_{(k)}$ 的相关系数

$$r_{jk} = \frac{s_{jk}}{\sqrt{s_{jj}} \sqrt{s_{kk}}} = \frac{s_{jk}}{s_j s_k} \quad j, k = 1, 2, \dots, n \quad (2-43)$$

r_{jk} 是无量纲的量, 总有 $r_{jj} = 1, |r_{jk}| \leq 1$ 。

称

$$\mathbf{R} = \begin{bmatrix} 1 & r_{12} & \cdots & r_{1n} \\ r_{21} & 1 & \cdots & r_{2n} \\ \vdots & \vdots & & \vdots \\ r_{n1} & r_{n2} & \cdots & 1 \end{bmatrix} \quad (2-44)$$

是观测数据的相关矩阵。相关矩阵 \mathbf{R} 是 n 维观测数据的最重要的数字特征, 它刻画了变量之间线性联系的密切程度。

(2) 总体参数

设 $(X_{(1)}, X_{(2)}, \dots, X_{(n)})$ 是 n 维总体, 其总体分布函数是 $F(x_1, x_2, \dots, x_n) = F(\mathbf{X})$, 其中 $\mathbf{X} = (x_1, x_2, \dots, x_n)^T$ 。在连续型总体时, 总体存在概率密度 $f(x_1, x_2, \dots, x_n) = f(\mathbf{X})$ 。

① 总体均值向量: 令 $\mu_i = E(X_{(i)})$, $i = 1, 2, \dots, n$, 则

$$\boldsymbol{\mu} = (\mu_1, \mu_2, \dots, \mu_n)^T \quad (2-45)$$

② 总体协方差矩阵

$$\Sigma = \text{Cov}(\mathbf{X}) = E[(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^T] = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2n} \\ \cdots & \cdots & & \cdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_{nn} \end{bmatrix} = (\sigma_{jk})_{n \times n} \quad (2-46)$$

其中 $\sigma_{jk} = \text{Cov}(X_{(j)}, X_{(k)}) = E[(X_{(j)} - \mu_j)(X_{(k)} - \mu_k)^T]$ 。特别地, 当 $j = k$ 时, $\sigma_{jj} = \sigma_j^2 = \text{Var}(X_{(j)})$ 。

③ 总体的分量 $X_{(j)}, X_{(k)}$ 的相关系数

$$\rho_{jk} = \frac{\sigma_{jk}}{\sqrt{\sigma_{jj}} \sqrt{\sigma_{kk}}} = \frac{\sigma_{jk}}{\sigma_j \sigma_k} \quad (2-47)$$

④ 总体的相关矩阵

$$\boldsymbol{\rho} = \begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1n} \\ \rho_{21} & 1 & \cdots & \rho_{2n} \\ \cdots & \cdots & & \cdots \\ \rho_{n1} & \rho_{n2} & \cdots & 1 \end{bmatrix} = (\rho_{jk})_{n \times n} \quad (2-48)$$

总有 $\rho_{jj} = 1, |\rho_{jk}| \leq 1$ 。

2.7.2 特征空间分布分析

1. 分布密度函数

设观测数据是由总体 \mathbf{X} 中取出的样本, 总体的分布函数是 $F(\mathbf{X})$ 。当 \mathbf{X} 为离散分布时, 总

体的分布可由概率分布刻画

$$p_i = P\{X = X_i\} \quad i = 1, 2, \dots$$

总体为连续分布时,总体的分布可由概率密度 $f(\mathbf{X})$ 刻画。几种常用一维连续总体分布的概率密度如下。

正态分布

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (2-49)$$

对数正态分布

$$P(x) = \begin{cases} \frac{1}{\sqrt{2\pi}\sigma(x-\theta)} \exp\left[-\frac{(\log(x-\theta)-\xi)^2}{2\sigma^2}\right], & x > \theta \\ 0, & \text{其他} \end{cases} \quad (2-50)$$

指数分布

$$P(x) = \begin{cases} \frac{1}{\sigma} \exp\left(-\frac{x-\theta}{\sigma}\right), & x > \theta \\ 0, & \text{其他} \end{cases} \quad (2-51)$$

Γ 分布(Gamma 分布)

$$P(x) = \begin{cases} \frac{1}{\Gamma(\alpha)\sigma} \left(\frac{x-\theta}{\sigma}\right)^{\alpha-1} \exp\left(-\frac{x-\theta}{\sigma}\right), & x > \theta \\ 0, & \text{其他} \end{cases} \quad (2-52)$$

2. 多维正态分布的性质

在进行模式识别方法的研究时,常用正态分布概率模型来抽取所需要的训练样本集和测试样本集,在数学上实现起来比较方便。

若 n 维总体 $\mathbf{X} = (\mathbf{X}_{(1)}, \mathbf{X}_{(2)}, \dots, \mathbf{X}_{(n)})^T$ 具有概率密度

$$\begin{aligned} P(\mathbf{X}) &= P(x_1, x_2, \dots, x_n) \\ &= \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{X}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{X}-\boldsymbol{\mu})\right\} \end{aligned} \quad (2-53)$$

则称 n 维总体服从 n 维正态分布。记为 $N_n(\boldsymbol{\mu}, \Sigma)$ 。记 $\mathbf{X} = (\mathbf{X}_{(1)}, \mathbf{X}_{(2)}, \dots, \mathbf{X}_{(n)})$, 则可证, n 维随机向量 \mathbf{X} 的均值向量为 $\boldsymbol{\mu}$, 协方差矩阵为 Σ 。多维正态分布的性质有以下几点。

(1) 参数 $\boldsymbol{\mu}$ 和 Σ 对分布的决定性

多元正态分布被均值向量 $\boldsymbol{\mu}$ 和协方差矩阵 Σ 所完全确定。由 $\boldsymbol{\mu} = E(\mathbf{X})$ 和 $\Sigma = E[(\mathbf{X}-\boldsymbol{\mu})(\mathbf{X}-\boldsymbol{\mu})^T]$ 可见, 均值向量 $\boldsymbol{\mu}$ 由 n 个分量组成, 协方差矩阵 Σ 由于其对称性故其独立元素只有 $n(n+1)/2$ 个, 所以, 多元正态分布由 $n + n(n+1)/2$ 个参数所完全确定。

(2) 不相关性等价于独立性

在数理统计中, 一般来说, 若两个随机变量 x_i 和 x_j 之间不相关, 并不意味着它们之间一定独立。下面给出不相关与独立的定义。

若 $E\{x_i x_j\} = E\{x_i\} E\{x_j\}$, 则定义随机变量 x_i 和 x_j 是不相关的。

若 $p(x_i x_j) = p(x_i) p(x_j)$, 则定义随机变量 x_i 和 x_j 是独立的。

从它们的定义中可以看出,独立性是比不相关性更强的条件,独立性要求 $p(x_i x_j) = p(x_i)p(x_j)$ 对于 x_i 和 x_j 都成立,而不相关性说的是两个随机变量的积的期望等于两个随机变量的期望的积,它反映了 x_i 和 x_j 总体的性质。若 x_i 和 x_j 相互独立,则它们之间一定不相关;反之则不一定成立。

对多维正态分布的任意两个分量 x_i 和 x_j 而言,若 x_i 和 x_j 互不相关,则它们之间一定独立。这就是说,在正态分布中不相关性等价于独立性。

(3) 边缘分布和条件分布的正态性

多维正态分布的边缘分布和条件分布仍然是正态分布。

(4) 线性变换的正态性

设 $\mathbf{X} \sim N_n(\boldsymbol{\mu}, \Sigma)$, 又 $\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{b}$, 其中 \mathbf{b} 是 n 维常向量, \mathbf{A} 是 $l \times n$ 矩阵, $\text{rank}(\mathbf{A}) = l$, 则

$$\mathbf{Y} \sim N_l(\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{A} \Sigma \mathbf{A}^T)$$

即 \mathbf{Y} 服从以 $\mathbf{A}\boldsymbol{\mu} + \mathbf{b}$ 为均值, 以 $\mathbf{A} \Sigma \mathbf{A}^T$ 为协方差矩阵的 l 维正态分布。

3. 多维正态分布总体参数的估计

在实际中,多维正态分布 $N(\boldsymbol{\mu}, \Sigma)$ 的参数 $\boldsymbol{\mu}$ 和 Σ 常常是未知的,需要通过样本来估计。

记 $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N$ 是从总体 \mathbf{X} 中取出的一个样本, 设总体的分布是连续型的, 分布密度函数为 $p(\mathbf{X}, \theta_1, \theta_2, \dots, \theta_k)$, 其中 $\theta_1, \theta_2, \dots, \theta_k$ 是待估计的未知参数, 对于给定的 $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N$, 使函数 $\prod_{i=1}^N p(\mathbf{X}_i, \theta_1, \theta_2, \dots, \theta_k)$ 达到最大值的 $\theta_1, \theta_2, \dots, \theta_k$, 应用它们分别作为 $\theta_1, \theta_2, \dots, \theta_k$ 的估计值。由于 $\ln \prod_{i=1}^N p(\mathbf{X}_i, \theta_1, \theta_2, \dots, \theta_k)$ 与 $\prod_{i=1}^N p(\mathbf{X}_i, \theta_1, \theta_2, \dots, \theta_k)$ 在同一点 $\theta_1, \theta_2, \dots, \theta_k$ 上达到最大值, 因此, 引入函数

$$L(\theta_1, \theta_2, \dots, \theta_k) = \ln \prod_{i=1}^N p(\mathbf{X}_i, \theta_1, \theta_2, \dots, \theta_k) = \sum_{i=1}^N \ln p(\mathbf{X}_i, \theta_1, \theta_2, \dots, \theta_k)$$

它称为似然函数, 只要解方程组

$$\frac{\partial L}{\partial \theta_i} = 0, (i = 1, 2, \dots, k) \quad (2-54)$$

就可以从中确定所要求的 $\theta_1, \theta_2, \dots, \theta_k$, 它们分别称为参数 $\theta_1, \theta_2, \dots, \theta_k$ 的最大似然估值。如果总体的分布是离散型的, 只要把上述似然函数中的 $p(x_i, \theta_1, \theta_2, \dots, \theta_k)$ 取为 $P(\mathbf{X} = \mathbf{X}_i)$ 就可以了。具体实现步骤如下。

① 设 $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N$ 是来自总体 $N_n(\boldsymbol{\mu}, \Sigma)$ 的简单随机样本, 则 $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N$ 的联合概率密度是 $\boldsymbol{\mu}, \Sigma$ 的函数。

② 构造似然函数:

$$\begin{aligned} L(\boldsymbol{\mu}, \Sigma) &= \prod_{i=1}^N \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left\{ -\frac{1}{2} (\mathbf{X}_i - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{X}_i - \boldsymbol{\mu}) \right\} \\ &= (2\pi)^{-\frac{Nn}{2}} |\Sigma|^{-\frac{N}{2}} \exp \left\{ -\frac{1}{2} \Sigma (\mathbf{X}_i - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{X}_i - \boldsymbol{\mu}) \right\} \end{aligned} \quad (2-55)$$

注意: 当得到样本观测值 $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N$ 后, $L(\boldsymbol{\mu}, \Sigma)$ 是 $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N$ 的函数。

③ 对总体 μ 和 Σ 的最大似然估计。在统计学中, μ, Σ 是未知的, 需要由样本观测值 X_1, X_2, \dots, X_N 估计。若 μ, Σ 作为 $X_{(1)}, X_{(2)}, \dots, X_{(n)}$ 的函数

$$\begin{aligned}\mu &= \hat{\mu}(X_{(1)}, X_{(2)}, \dots, X_{(n)}) \\ \Sigma &= \hat{\Sigma}(X_{(1)}, X_{(2)}, \dots, X_{(n)})\end{aligned}$$

满足 $L(\hat{\mu}, \hat{\Sigma}) = \max_{\mu, \Sigma} L(\mu, \Sigma)$, 则 $\hat{\mu}, \hat{\Sigma}$ 称为 μ, Σ 的最大似然估计。

对任意 N 元总体, 均值向量 \bar{X} 、协方差矩阵 S 是总体均值向量 μ 、总体协方差矩阵 Σ 的估计。而对 N 元正态总体, Σ 的最大似然估计为

$$\hat{\Sigma} = \frac{N-1}{N} S \quad (2-56)$$

当 N 较大时, $\hat{\Sigma} \approx S$ 。因 S 是 Σ 的无偏估计, 通常仍以 S 作为 Σ 的估计。

2.8 手写数字特征提取与分析

2.8.1 手写数字特征提取

本书以手写数字作为模式分类的实例, 重点介绍模式识别理论与实现方法, 说明各种算法是否有效。

对数字识别特征提取可以有多种方法, 有的分析从框架的左边框到数字之间的距离变化, 反映了不同数字的不同形状, 这可以用来作为数字分类的依据, 如图 2-5 所示; 另外一种方法则在每个数字图形上定义一个 $N \times N$ 模板, 将每个样品的长度和宽度 N 等分, 平均有 $N \times N$ 个等份, 对每一份内的像素个数进行统计, 除以每一份的面积总数, 即得特征初值。

如图 2-6(a) 所示。首先找到每个手写样品的起始位置, 在此附近搜索该样品的宽度和高度, 将每个样品的长度和宽度 5 等分, 构成一个 5×5 均匀小区域; 对于每一小区域内的黑像素个数进行统计, 除以该小区域的面积总数, 即得特征值, 如图 2-6(b) 所示。当然读者可以根据需要进行修改, N 值越大, 模板也越大, 特征越多, 区分不同的物体能力越强, 但同时计算量增加, 运行等候的时间增长, 所需要的样本库也成倍增加, 一般样本库的个数为特征数的 5~10 倍, 这里特征总数为 $5 \times 5 = 25$, 每一种数字就需要至少 125 个标准样本, 10 个数字需要 1250 个标准样本, 可想而知数目已经不少了。如果值过小, 不利于不同物体间的区别。

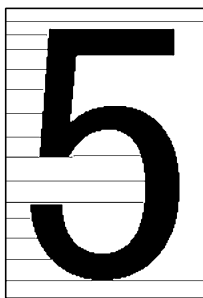


图 2-5 距离变化提取特征法

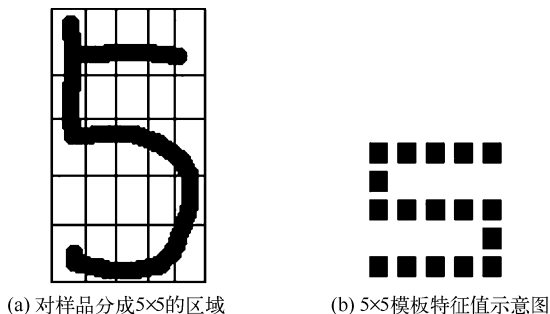


图 2-6 5×5 模板提取特征法

对于手写数字提取模板特征的好处是,针对同一形状、不同大小的样品得到的特征值相差不大。有能力对同一形状、不同大小的样品视为同类,因此这里要求物体至少在宽度和长度上大于5个像素,太小则无法正确分类。当然读者可以根据需要进行修改,值越大,模板也越大,特征越多,区分不同的物体能力越强,但同时计算量增加,运行等候的时间增长,所需样品库也要成倍增加。

在本书配套程序中,读者可以手写一个数字,然后应用不同的模式识别算法实现对数字的识别。识别过程如下。

(1) 手写数字

在界面上手写一个数字,按“清除”键后可重新书写。

(2) 手写数字的特征提取

- ① 搜索数据区,找出手写数字的上、下、左、右边界。
- ② 将数字区域平均分为 5×5 的小区域。
- ③ 计算 5×5 的每一个小区域中黑像素所占比例,第一行的5个比例值保存到特征的前5个,第二行对应着特征的6~10个,以此类推。

(3) 建立训练集特征库

分类器的设计方法属于监督学习法。在监督学习过程中,为了能够对未知事物进行分类,必须输入一定数量的样本,构建训练集,而且这些样本的类别已知,提取这些样本的特征,构造分类器,然后对任何未知类别的模式识别。读者可以直接书写数字,单击“选择类本”下拉列表框,为手写的数字选择其对应的类别。单击“保存样本”按钮,根据提示,将样本保存到样本库的首位,如图2-7所示。



图2-7 保存样本示意图

(4) 通过对话框查看样本库样本个数

单击“选择类别”下拉列表框,选择一个类别。然后单击“查看样本特征”按钮,可以在Matlab命令窗口中查看每个类别的样本特征值。

(5) 分类识别

用分类器判别样品类型。

在分类程序中,样本库训练集的特征值是程序开发人员按照自己手写数字习惯来建立的,因此,会造成对读者手写的数字分类有误的情况。为了避免此类情况发生,我们把每次添加的手写数字放在样本训练集的首位,读者可以尽量多写一些数字以使程序适应您的书写样式。

2.8.2 手写数字特征空间分布分析

在工程上的许多问题中,统计数据往往满足正态分布规律。正态分布简单,参量少,分析方便,是一种适宜的数学模型。

尽管不同的人手写数字形状有所区别,但在特征空间中对某一类的特征进行观察时,这些手写的数字较多地分布在这一类的均值附近,远离均值点较少,因此用正态分布作为这一类的概率模型是合理的。

要了解手写数字特征空间分布情况,需要根据现有的训练样本集,对总体参数做点估计和区间估计,然后假设样本的总体分布函数,对总体分布函数进行统计假设检验。由于对手写数字提取了 25 维特征,为了简化分布分析过程,对手写数字特征空间水平投影降维为 5 维。在此基础上进行主成分分析,提取第一主成分分量作为有效特征,进行一维正态分布分析。

1. 手写数字总体参数的估计

(1) 总体参数的点估计

采用最大似然法对手写数字总体参数进行点估计。

由于假设手写数字特征分布遵从正态分布 $N(\mu, \sigma^2)$, 但总体参数 $\theta(\mu, \sigma^2)$ 未知, 对于正态总体的参数估计现在要用总体的 N 次观测值 x_1, x_2, \dots, x_N , 求 μ, σ^2 的最大似然估值。

一元总体的分布密度函数为

$$p(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (2-57)$$

似然函数为

$$L(\mu, \sigma) = -\frac{1}{2\pi\sigma^2} \sum_{i=1}^N (x_i - \mu)^2 - n \ln \sigma - \frac{n}{2} \ln 2 \quad (2-58)$$

解方程组

$$\begin{cases} \frac{\partial L}{\partial \mu} = 0 \\ \frac{\partial L}{\partial \sigma} = 0 \end{cases}$$

得

$$\begin{aligned} \mu &= \bar{x} = \frac{1}{N} \sum_{i=1}^N x_i \\ \sigma^2 &= \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 \end{aligned}$$

容易检验 μ, σ^2 确实使 $L(\mu, \sigma)$ 取到最大值, 因此它们分别是 μ, σ^2 的最大似然估值。

(2) 估值好坏的判别标准

① 无偏性。如果参数 θ 的估值 $\hat{\theta}_N(x_1, x_2, \dots, x_N)$ 满足关系式

$$E\hat{\theta}_N = \theta$$

则称 $\hat{\theta}_N$ 是 θ 的无偏估值。

② 有效性。如果 $\hat{\theta}$ 和 $\hat{\theta}'$ 都是参数 θ 的无偏估值

$$D\hat{\theta} \leq D\hat{\theta}'$$

则称 $\hat{\theta}$ 比 $\hat{\theta}'$ 有效。进一步,如果固定样本的容量 N ,使 $D\hat{\theta}' = \text{极小值的无偏估值}$ 就成为 θ 的有效估值。

③ 一致性。如果对任意给定的正数 ε ,总有

$$\lim_{N \rightarrow \infty} P(|\hat{\theta}_N - \theta| > \varepsilon) = 0$$

则称 θ 的估值 $\hat{\theta}_N$ 是一致的。当

$$\lim_{N \rightarrow \infty} E|\hat{\theta}_N - \theta|^r = 0$$

对某 $r \geq 0$ 成立时, $\hat{\theta}_N$ 是 θ 的一致估值。

(3) 总体参数的区间估计

在一次试验中,概率很小(接近于零)的事件认为是实际上不可能发生的事件;而概率接近于1的事件认为是实际上必然发生的事件。

对总体参数 $\theta(\mu, \sigma^2)$ 进行区间估计(即区间参数的取值范围)时,如果对于预先给定的很小的概率 α ,能找到一个区间 (θ_1, θ_2) ,使得

$$P(\theta_1 < \theta < \theta_2) = 1 - \alpha$$

那么称区间 (θ_1, θ_2) 为参数 θ 的置信区间, θ_1 和 θ_2 称为置信限(或临界值); $\theta \leq \theta_1$ 和 $\theta \geq \theta_2$ 称为否定域;概率 α 称为显著性水平, $1 - \alpha$ 称为置信水平(或置信概率)。

假设总体遵从正态分布 $N(\mu, \sigma^2)$ 。对于预先给定的显著性水平 α ,可用一个样本 $\{x_1, x_2, \dots, x_N\}$ 的均值 \bar{x} 和标准差 s 来估计总体的均值 μ 和方差 σ^2 的置信区间,估计方法如表2-2所示。

表2-2 (小样本)置信区间的估计方法

样 本 情 况	总体参数 μ 或 σ^2 的置信区间	与置信区间有关的 $K_\alpha, t_\alpha, \chi_\alpha^2$ 与 F_α 的确定
小样本已知 总体方差	$\mu \in \left(\bar{x} - \frac{K_{\frac{\alpha}{2}}\sigma}{\sqrt{N}}, \bar{x} + \frac{K_{\frac{\alpha}{2}}\sigma}{\sqrt{N}} \right)$	$\int_{-\frac{K_{\frac{\alpha}{2}}}{2}}^{\frac{K_{\frac{\alpha}{2}}}{2}} \frac{1}{\sqrt{2\pi}} e^{-\frac{v^2}{2}} dv = 1 - \alpha$ 查正态分布表
小样本总体 方差未知	$\mu \in \left(\bar{x} - \frac{t_{\frac{\alpha}{2}}s}{\sqrt{N}}, \bar{x} + \frac{t_{\frac{\alpha}{2}}s}{\sqrt{N}} \right)$	$\int_{-t_{\frac{\alpha}{2}}}^{t_{\frac{\alpha}{2}}} t(n-1) dv = 1 - \alpha$ 查 t 分布表(自由度为 $n-1$)
小样本已知 总体均值	$\sigma^2 \in \left(\frac{1}{\chi_{\frac{\alpha}{2}}^2} \sum_{i=1}^N (x_i - \mu)^2, \frac{1}{\chi_{\frac{\alpha}{2}}^2} \sum_{i=1}^N (x_i - \mu)^2 \right)$	$\int_{\chi_{\frac{\alpha}{2}}^2}^{\chi_{\frac{\alpha}{2}}^2} \chi^2(n) dv = 1 - \alpha$ 查 χ^2 分布表(自由度为 n)
小样本总体 均值未知	$\sigma^2 \in \left(\frac{N-1}{\chi_{\frac{\alpha}{2}}^2} s^2, \frac{N-1}{\chi_{\frac{\alpha}{2}}^2} s^2 \right)$	$\int_{\chi_{\frac{\alpha}{2}}^2}^{\chi_{\frac{\alpha}{2}}^2} \chi^2(n-1) dv = 1 - \alpha$ 查 χ^2 分布表(自由度为 $n-1$)

2. 总体分布函数的 χ^2 检验法

(1) χ^2 检验法

χ^2 检验法是基于对数据按其取值范围进行分组后计算频数的基础上,考虑每个区间的实际频数 $\{v_i\}$ 与理论频数 $\{p_i\}$ 的差异做出判断。它使用的统计量

$$\chi^2 = \sum_{i=1}^l \frac{(v_i - Np_i)^2}{Np_i} \quad (2-59)$$

其中, N 为样本数据的容量, l 是分组数, p_i 根据原假设指定的分布求得。

(2) 检验步骤

假设 x 的分布函数为 $F(x)$,这时,相应的假设检验问题为

$$H_0: F(x) \equiv F_0(x) \leftrightarrow H_1: F(x) \text{ 不是 } F_0(x)$$

分两种情况进行统计假设检验。

① 设 $F_0(x) = F_0(x, \theta_1, \theta_2, \dots, \theta_k)$ 为已知类型的分布函数, $\theta_1, \theta_2, \dots, \theta_k$ 为参数(已知或部分已知), x_1, x_2, \dots, x_N 为总体 X 的样本,把实轴 $(-\infty, \infty)$ 分成 l 个不相交的区间: $(c_i, c_{i+1}]$ ($i=1, 2, \dots, l$), $c_1 = -\infty, c_{l+1} = \infty$, 其中 $(c_l, c_{l+1}]$ 理解成 (c_l, ∞) 。

理论频数记为

$$p_i = F_0(c_{i+1}) - F_0(c_i) = P(c_i \leq x \leq c_{i+1})$$

x 的样本 $\{x_1, x_2, \dots, x_N\}$ 落在区间 $(c_i, c_{i+1}]$ 的个数为 v_i (经验频数),根据式(2-59)计算统计量 χ^2 。设 k 是原假设指定的分布类中的待估参数的个数,遵从自由度为 $l-k-1$ 的 χ^2 分布,应用 χ^2 检验法便可检验假设 $H_0: F(x) = F_0(x)$ 是否可信。

若原假设成立, χ^2 的值应比较小,所以当 χ^2 取大的值时是极端情形。

例如,原假设是正态分布, $F_0(x) = \int_{-\infty}^x p(t, \mu, \sigma) dt = \int_{-\infty}^x \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt$, 此时 $k=2$, 统计学

研究表明:当样本容量 N 充分大且原假设 H_0 为真时, χ^2 统计量近似服从自由度为 $l-k-1$ 的 χ^2 分布,即

$$\chi^2: \chi^2(l-k-1)$$

给定显著水平 α ,设由样本观测值算得的 χ^2 值是 χ_0^2 。则当 $\chi_0^2 > \chi_\alpha^2(l-k-1)$ 时,拒绝 H_0 ;否则,接受 H_0 。

② $F_0(x)$ 的参数全部或部分未知。设 $F_0(x)$ 有 l 个参数 $\theta_{j_1}, \theta_{j_2}, \dots, \theta_{j_l} (j \leq k)$ 未知,可先用最大似然估计法定出这 l 个参数的估值,把这些估值就当做 $F_0(x)$ 的相应参数,于是类似①的情形可计算理论概率,再计算经验频数,那么按式(2-59)计算统计量。当 N 很大时遵从自由度为 $l-k-1$ 的 χ^2 分布。

应用 χ^2 检验法便可检验假设 $H_0: F(x) = F_0(x)$ 是否可行。

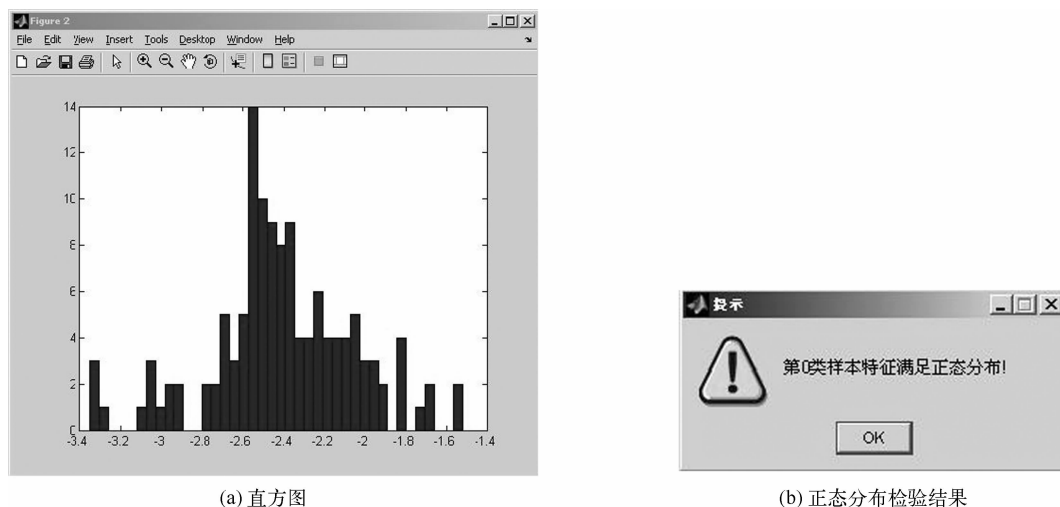


图 2-8 数字 0 特征空间分布检验效果图

本章小结

在实际应用中,信息采集对象多数是多特征,高噪声,非线性的数据集。人们只能尽量多列一些可能有影响的因素,在样本数不是很多的情况下,用很多特征进行分类器设计,无论从计算的复杂程度还是就分类器性能来看都是不适宜的。因此研究如何把高维特征空间压缩到低维特征空间就成了一个重要的课题。特征选择与提取是模式识别的基础环节,从样本采集到建立满意的特征和特征库,需要经过多次反复试验。本章将多元统计分析 with 模式识别特征处理理论相结合,介绍了样本特征库初步分析方法,特征筛选处理方法,特征选择及搜索算法,特征评估方法,主成分分析方法、特征提取方法;并介绍了特征空间统计量的描述及特征空间分布分析方法。

习题 2

1. 简述样本数量与特征数目的关系,若采集的手写数字各个类别的样本数目少于特征数目,能否对手写数字分类?
2. 某一个特征与目标的相关系数为 0,该特征是否删除?
3. 简述特征选择搜索算法。
4. 简述几种常用的特征评估方法。
5. 简述主成分分析的实现方法。
6. 简述总体参数的点估计方法。
7. 简述总体分布函数的统计假设检验方法。

第 3 章 模式相似性测度

本章要点:

- ☑ 模式相似性测度的基本概念
- ☑ 距离测度分类法

3.1 模式相似性测度的基本概念

模式识别最基本的研究问题是样品与样品之间或类与类之间相似性测度问题。判断样品之间的相似性常采用近邻准则,即将待分类样品与标准模板进行比较,看跟哪个模板匹配程度更好些,从而确定待测试样品的分类。近邻法则在原理上属于模板匹配。它将训练样品集中的每个样品都作为模板,用测试样品与每个模板做比较,看与哪个模板最相似(即为近邻),就按最近似的模板的类别作为自己的类别。计算模式相似性测度有欧式距离、马氏距离、夹角余弦距离、Tanimoto 测度等多种距离算法。依照近邻准则进行分类通常有两种计算方法,一是通过与样本库所有样品特征分别作相似性测度,找出最接近的样品,取该样品所属类别作为待测试样品的类别。另一种方法是与样本库中不同类别的中心或重心做相似性测度,找出最接近类的中心,以该类作为待测样品的类别。例如,A 类有 10 个训练样品,因此有 10 个模板,B 类有 8 个训练样品,就有 8 个模板。任何一个待测试样品在分类时与这 18 个模板都算一算相似度,如最相似的那个近邻是 B 类中的一个,就确定待测试样品为 B 类,否则为 A 类。另一种方法是:分别求出 A 类和 B 类的中心,待测试样品分别与这两个中心做相似性测度,与哪个类的中心最接近,则待测样品归为该类的。

原理上说近邻法是最简单的。但是近邻法有一个明显的缺点就是计算量大,存储量大,要存储的模板很多,当每个测试样品要对每个模板计算一次相似度时,所需的计算时间相对其他方法多一些。

1. 样品与样品之间的距离

设有两个样品 \mathbf{X}_i 、 \mathbf{X}_j 的特征向量分别为

$$\mathbf{X}_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{in} \end{pmatrix} = (x_{i1}, x_{i2}, \dots, x_{in})^T, \quad \mathbf{X}_j = \begin{pmatrix} x_{j1} \\ x_{j2} \\ \vdots \\ x_{jn} \end{pmatrix} = (x_{j1}, x_{j2}, \dots, x_{jn})^T$$

这两个样品可能在同一个类中,如图 3-1(a) 所示,也可能在不同的类中,如图 3-1(b) 所示。因此,可以计算同一个类内样品与样品之间的距离,也可以计算属于不同类样品与样品之间的距离。

样品与样品间的距离计算有五种方法,分别是欧氏距离法、马氏距离法、夹角余弦距离法、

二值夹角余弦法和具有二值特征的 Tanimoto 测度,如表 3-1 所示。

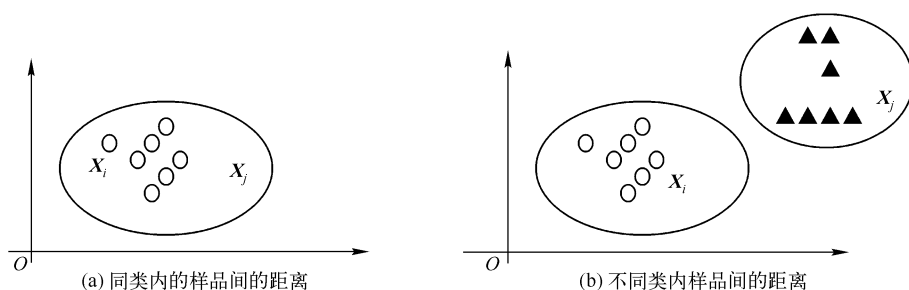


图 3-1 样品间的距离示意图

表 3-1 样品间的距离计算公式

计算距离方法	样品间距离计算公式	说 明
欧氏距离	$D_{ij}^2 = (\mathbf{X}_i - \mathbf{X}_j)^T (\mathbf{X}_i - \mathbf{X}_j) = \ \mathbf{X}_i - \mathbf{X}_j\ ^2$ $= \sum_{k=1}^n (x_{ik} - x_{jk})^2$	D_{ij} 越小,则两个样品距离越近,就越相似
马氏距离	$D_{ij}^2 = (\mathbf{X}_i - \mathbf{X}_j)^T \mathbf{S}^{-1} (\mathbf{X}_i - \mathbf{X}_j)$ $\mathbf{S} = \frac{1}{N-1} \sum_{i=1}^N (\mathbf{X}_i - \bar{\mathbf{X}})(\mathbf{X}_i - \bar{\mathbf{X}})^T$ $\bar{\mathbf{X}} = \frac{1}{N} \sum_{i=1}^N \mathbf{X}_i$	D_{ij} 越小,则两个样品距离越近,就越相似
夹角余弦距离	$S(\mathbf{X}_i, \mathbf{X}_j) = \cos\theta = \frac{\mathbf{X}_i^T \mathbf{X}_j}{\ \mathbf{X}_i\ \cdot \ \mathbf{X}_j\ }$	S 值越大,则相似度越大
特征是二值时的夹角余弦	$S(\mathbf{X}_i, \mathbf{X}_j) = \cos\theta = \frac{\mathbf{X}_i^T \mathbf{X}_j}{\sqrt{(\mathbf{X}_i^T \mathbf{X}_i)(\mathbf{X}_j^T \mathbf{X}_j)}}$	要求 $\mathbf{X}_i, \mathbf{X}_j$ 向量的各个特征都是以二值(0 或 1)表示, S 越大越相似
具有二值特征的 Tanimoto 测度	$S(\mathbf{X}_i, \mathbf{X}_j) = \frac{\mathbf{X}_i^T \mathbf{X}_j}{\mathbf{X}_i^T \mathbf{X}_i + \mathbf{X}_j^T \mathbf{X}_j - \mathbf{X}_i^T \mathbf{X}_j}$	要求 $\mathbf{X}_i, \mathbf{X}_j$ 向量的各个特征都以二值(0 或 1)表示, S 越大越相似

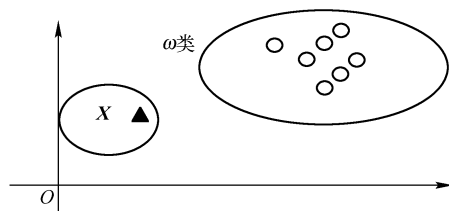


图 3-2 样品与类之间的距离

2. 样品与类之间的距离

如图 3-2 所示,形象地表示出样品与类之间的距离。 ω 是代表某类样品的集合, ω 中有 N 个样品, \mathbf{X} 是某一个待测样品。

样品与类之间距离的计算方法有两种。

① 计算该样品到 ω 类内各个样品之间的距离,将这些距离求和,取平均值作为样品与类之间的距离。样品与类之间的距离可描述为

$$\overline{D^2}(\mathbf{X}, \omega) = \frac{1}{N} \sum_{i=1}^N D^2(\mathbf{X}, \mathbf{X}_i^{(\omega)}) = \frac{1}{N} \sum_{i=1}^N \sum_{k=1}^n |x_k - x_{ik}^{(\omega)}|^2 \quad (3-1)$$

② 计算 ω 类的中心点 $\mathbf{M}^{(\omega)}$,以 ω 中的所有样品特征的平均值作为类中心,然后计算待测样品 \mathbf{X} 到 ω 的中心点 $\mathbf{M}^{(\omega)}$ 的距离。

$$D^2(\mathbf{X}, \omega) = D^2(\mathbf{X}, \mathbf{M}^{(\omega)}) = \sum_{k=1}^n |x_k - m_k^{(\omega)}|^2 \quad (3-2)$$

本书实例均采用式(3-2)作为样品与类之间的距离计算公式。

3. 类内距离

类内距离是指同一个类内任意样品之间距离之和的平均值。如图 3-3 所示,类内点集 $\{\mathbf{X}_i, i = 1, 2, \dots, N\}$ 各点之间的内部距离平方为 $\overline{D^2(\{\mathbf{X}_i\}, \{\mathbf{X}_j\})}$, $(i, j = 1, 2, \dots, N, i \neq j)$, 从集内一固定点 \mathbf{X}_i 到所有其他的 $N - 1$ 个点 \mathbf{X}_j 之间的距离平方是 $\overline{D^2(\mathbf{X}_i, \{\mathbf{X}_j\})} = \frac{1}{N-1} \sum_{j=1, j \neq i}^N \sum_{k=1}^n (x_{ik} - x_{jk})^2$ 。同样道理,取 ω 内所有 N 个点的平均距离以表示其类内距离:

$$\overline{D^2(\{\mathbf{X}_i\}, \{\mathbf{X}_j\})} = \frac{1}{N} \sum_{i=1}^N \left[\frac{1}{N-1} \sum_{j=1, j \neq i}^N \sum_{k=1}^n (x_{ik} - x_{jk})^2 \right] = \frac{1}{N(N-1)} \sum_{i=1}^N \sum_{j=1, j \neq i}^N \sum_{k=1}^n (x_{ik} - x_{jk})^2 \quad (3-3)$$

4. 类与类之间的距离

设有两个类 ω_i, ω_j , 如图 3-4 所示, 计算类与类之间的距离有多种方法, 例如, 最短距离法、最长距离法、重心法和平均距离法等, 如表 3-2 所示。

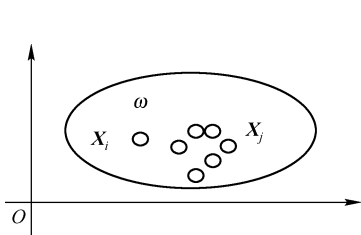


图 3-3 ω 类内的距离

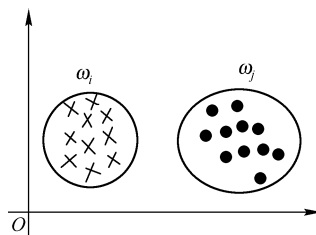


图 3-4 类间距离

表 3-2 类间的距离

距离方法	距离定义	说明
最短距离法	规定两个类间相距最近的两个点之间的距离, 为两类的距离	$D_{i,j} = \min(d_{ij})$ $d_{ij} = \ \mathbf{X}_i - \mathbf{X}_j\ , \mathbf{X}_i \in \omega_i, \mathbf{X}_j \in \omega_j$
最长距离法	规定两个类间相距最远的两个点之间的距离, 为两类的距离	$D_{ij} = \max(d_{ij})$ $d_{ij} = \ \mathbf{X}_i - \mathbf{X}_j\ , \mathbf{X}_i \in \omega_i, \mathbf{X}_j \in \omega_j$
重心法	求各类中所有样品的平均值作为类的重心, 用两类的重心间的距离作为两类的距离	$D_{i,j} = \ \overline{\mathbf{X}^{(\omega_i)}} - \overline{\mathbf{X}^{(\omega_j)}}\ $ $\overline{\mathbf{X}^{(\omega_i)}} = \frac{1}{N_i} \sum_{\mathbf{X} \in \omega_i} \mathbf{X} \quad \overline{\mathbf{X}^{(\omega_j)}} = \frac{1}{N_j} \sum_{\mathbf{X} \in \omega_j} \mathbf{X}$ N_i, N_j 分别是 ω_i, ω_j 类中样品的个数
平均距离法	计算两类之间所有样品的距离, 求和, 取距离的平均值作为两类间的距离	$D_{i,j} = \frac{1}{N_i N_j} \sum_{\substack{\mathbf{X}_i \in \omega_i \\ \mathbf{X}_j \in \omega_j}} \ \mathbf{X}_i - \mathbf{X}_j\ $

3.2 距离测度分类法

3.2.1 模板匹配法

1. 理论基础

最简单的识别方法就是模板匹配,就是把未知样品和一个标准模板相比,看它们是否相同或相似。下面讨论两类别和多类别的情况。

(1) 两类别

设有两个标准样品模板为 A 和 B,其特征向量为 n 维特征: $\mathbf{X}_A = (x_{A1}, x_{A2}, \dots, x_{An})^T$ 和 $\mathbf{X}_B = (x_{B1}, x_{B2}, \dots, x_{Bn})^T$ 。任何一个待识别的样品 \mathbf{X} ,其特征向量为: $\mathbf{X} = (x_1, x_2, \dots, x_n)^T$,那么,它是 A,还是 B 呢?

用模板匹配方法来识别,若 $\mathbf{X} = \mathbf{X}_A$,则该样品为 A,若 $\mathbf{X} = \mathbf{X}_B$,则该样品为 B。怎样知道 $\mathbf{X} = \mathbf{X}_A$,还是 $\mathbf{X} = \mathbf{X}_B$ 呢? 最简单的识别方法就是利用距离来判别。如果 \mathbf{X} 距离 \mathbf{X}_A 比距离 \mathbf{X}_B 近,则 \mathbf{X} 属于 \mathbf{X}_A ,否则属于 \mathbf{X}_B 。这就是最小距离判别法。

任意两点 \mathbf{X}, \mathbf{Y} 之间的距离:

$$d(\mathbf{X}, \mathbf{Y}) = \left[\sum_{i=1}^n (x_i - y_i)^2 \right]^{\frac{1}{2}} \quad (3-4)$$

根据距离远近可作为判据,构成距离分类器,其判别法则为

$$\begin{cases} d(\mathbf{X}, \mathbf{X}_A) < d(\mathbf{X}, \mathbf{X}_B) \Rightarrow \mathbf{X} \in A \\ d(\mathbf{X}, \mathbf{X}_A) > d(\mathbf{X}, \mathbf{X}_B) \Rightarrow \mathbf{X} \in B \end{cases}$$

(2) 多类别

设有 M 个类别: $\omega_1, \omega_2, \dots, \omega_M$ 。每类由若干个向量表示,如 ω_i 类,有

$$\mathbf{X}_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ \vdots \\ x_{in} \end{pmatrix}$$

对于任意被识别的样品 \mathbf{X} ,有

$$\mathbf{X} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix}$$

计算距离 $d(\mathbf{X}_i, \mathbf{X})$,若存在某一个 i ,使

$$d(\mathbf{X}_i, \mathbf{X}) < d(\mathbf{X}_j, \mathbf{X}), j = 1, 2, \dots, M, i \neq j$$

即到某一个样品最近,则 $\mathbf{X} \in \omega_i$ 。

$$\begin{aligned} d(X, X_i) &= |X - X_i|^2 = (X - X_i)^T (X - X_i) \\ &= X^T X - X^T X_i - X_i^T X + X_i^T X_i \\ &= X^T X - (X^T X_i + X_i^T X - X_i^T X_i) \end{aligned} \quad (3-5)$$
$$d_i(X) = X^T X_i + X_i^T X - X_i^T X_i \quad (3-6)$$

```
% 函数名称:neartemplet( )
% 参数:sample:待识别样品特征
% 返回值: y:待识别样品所属类别
% 函数功能:按照模板匹配法计算待测样品与样品库中的样品相似度
function y = neartemplet( sample );
    clc;
    load templet pattern;% 加载样品库
    d=0;% 距离
    min = [ inf,0 ];
    for i = 1:10
        for j = 1:pattern(i). num
            % 计算待测样品与样品库样本间的最小距离
            d = sqrt( sum( ( pattern(i). feature( :,j) - sample' ). ^2 ) );
            % 求最小距离及其类号
            if min(1) > d
                min(1) = d;
                min(2) = i-1 ;
            end
        end
    end
    % 输出类别
    y = min(2) ;
```

4. 效果图

运行效果如图 3-5 所示。



图 3-5 模板匹配法

3.2.2 基于 PCA 的模板匹配法

在使用模板匹配法之前,先对特征进行主成分分析。按照一定贡献值,提取前 m 个主分量,用较低维数的特征来进行分类。

1. 实现步骤

- ① 选取各类全体样本组成矩阵 $X_{n \times N}$,待测样品为 $X_{n \times 1}$ 。
- ② 计算 $X_{n \times N}$ 的协方差矩阵 $S_{n \times n}$ 。
- ③ 计算 $S_{n \times n}$ 的特征值 $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ 和特征向量 $C_{n \times n}$ 。
- ④ 根据一定的贡献率,选取 $C_{n \times n}$ 的前 m 列,构成 $C_{n \times m}$ 。
- ⑤ 计算样本库样本主成分 $X_{m \times N} = C_{n \times m}^T X_{n \times N}$ 和样品主成分 $X_{m \times 1} = C_{n \times m}^T X_{n \times 1}$ 。
- ⑥ 采用模板匹配法进行多类别分类。

2. 编程代码

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 函数名称:pcanearmtemplet( )
% 参数:sample:待识别样品特征
% 返回值: y:待识别样品所属类别
% 函数功能:按照使用 PCA 的模板匹配法计算待测样品与样品库中的样品相似度
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = pcanearmtemplet( sample );
    clc;
    load templet pattern;
    % 对样品和样品库进行主成分分析
    [pcapat,pcasamp] = pcapro( sample );
    temp = 0;

```

```
for i = 1 : 10
    pattern(i).feature = pcapat( :, temp + 1 : temp + pattern(i).num );
    temp = temp + pattern(i).num;
end
d = 0 ; % 距离
min = [ inf , 0 ] ;
for i = 1 : 10
    for j = 1 : pattern(i).num
        % 计算待测样品与样品库样品间的最小距离
        d = sqrt( sum( ( pattern(i).feature( :, j) - pcasamp ). ^ 2 ) ) ;
        % 求最小距离及其类号
        if min(1) > d
            min(1) = d;
            min(2) = i-1;
        end
    end
end
% 输出类别
y = min(2);
%% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %%
% 函数名称:pcapro()
% 参数:sample:待识别样品特征
% 返回值:y1:样品库样品经主成分分析后的主分量矩阵;y2:待识别样品经主成分分析后的
% 主分量向量
% 函数功能:对样品库和待测样品用主成分分析法进行降维
%% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %% %%
function [y1,y2] = pcapro(sample)
load templet pattern;% 加载样品库
mixedsig = [];
suml = 0;
% 将所有类别的所有样品合并到 mixedsig
for i = 1 : 10
    suml = suml + pattern(i).num;
    mixedsig = [ mixedsig pattern(i).feature ];
end
[Dim, NumofSampl] = size(mixedsig); % Dim 为特征数,NumofSampl 为样品总个数
dsig_cov = cov(mixedsig'); % 求 mixedsig 的协方差矩阵
% 利用 pcacov() 函数求得从大到小排好序的协方差矩阵的特征值 latent 和相应特征向量 pc
[pc,latent,tspare] = pcacov(dsig_cov);
temp = 0; con = 0; m = 0;
% 根据贡献率取舍特征向量
sum2 = sum(latent);
for i = 1 : 25
```

```

if( con < 0.9)
    temp = temp + latent(i);
    con = temp/sum2;
    m = m + 1;
else
    break;
end
end
pc(:, m + 1:25) = [];
% 求待测样品主成分
x = sample * pc;
% 求样本库品成分
y = mixedsig' * pc;
y1 = y';
y2 = x';

```

3. 效果图

运行效果如图 3-6 所示。



(a) 待测样品



(b) 分类结果

图 3-6 使用 PCA 的最邻近模板匹配法

3.2.3 基于类中心的欧式距离法分类

1. 理论基础

设有 M 个类别: $\omega_1, \omega_2, \dots, \omega_M$ 。每类有 N_i 个样品, 如 ω_i 类, 可表示为 $\mathbf{X}^{(\omega_i)} = (\mathbf{X}_1^{(\omega_i)}, \mathbf{X}_2^{(\omega_i)}, \mathbf{X}_3^{(\omega_i)}, \dots, \mathbf{X}_{N_i}^{(\omega_i)})^T$ 。对于任意待识别的样品 $\mathbf{X} = (x_1, x_2, x_3, \dots, x_n)$, 计算距离 $d^2(\mathbf{X}, \omega_i) = (\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}})^T (\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}})$, 其中 $\overline{\mathbf{X}^{(\omega_i)}}$ 为第 i 类的类中心。比较 \mathbf{X} 到各类的距离若满足下式

$$d(\mathbf{X}, \overline{\mathbf{X}^{\omega_i}}) < d(\mathbf{X}, \overline{\mathbf{X}^{\omega_j}}) \quad j = 1, 2, \dots, M, i \neq j$$

则 $\mathbf{X} \in \omega_i$ 。

2. 实现步骤

- ① 选取某一类样本 \mathbf{X} 。

- $$d^2(\mathbf{X}, \omega_i) = (\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}})^T (\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}})$$

- ### 3. 编程代码

4. 效果图

运行效果如图 3-7 所示。


```

% 求协方差矩阵
for i = 1:10
    for j = 1:pattern(i). num
        c = c + 1;
        pdata(:,c) = pattern(i). feature(:,j);
    end
end
% 求特征间的协方差矩阵及其逆矩阵
s_cov = cov(pdata');
s_inv = inv(s_cov);
d = 0; % 距离
p = []; % 各类别的代表点
dmin = [inf, 0];
% 求各类别中值点
for i = 1:10
    temp = mean(pattern(i). feature');
    p(:,i) = temp';
end
for i = 1:10
    % 计算待测样品与样品库样品间的马氏距离
    d = (sample' - p(:,i))' * s_inv * (sample' - p(:,i));
    % 求最小距离及其类号
    if dmin(1) > d
        dmin(1) = d;
        dmin(2) = i - 1;
    end
end
% 输出类别
y = dmin(2);

```

4. 效果图

运行效果如图 3-8 所示。



(a) 待测样品



(b) 分类结果

图 3-8 使用马氏距离法识别结果

3.2.5 夹角余弦距离分类

1. 实现步骤

- ① 待测样品 \mathbf{X} 与训练集里每个样品 \mathbf{X}_i 的距离采用下式计算。

$$S(\mathbf{X}, \mathbf{X}_i) = \cos\theta = \frac{\mathbf{X}^T \mathbf{X}_i}{\|\mathbf{X}\| \cdot \|\mathbf{X}_i\|}$$

- ② 循环计算待测样品和训练集中各已知样品之间的距离,找出距离待测样品最近的已知样品,该已知样品的类别就是待测样品的类别。

2. 编程代码

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% 函数名称:anglecos()
% 参数:sample:待识别样品特征
% 返回值:y:待识别样品所属类别
% 函数功能:按照夹角余弦距离法计算待测样品与样品库中的样品相似度
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
function y = anglecos(sample);

    clc;

    load templet pattern;
    d=0;%距离
    dmax=[-inf,0];
    for i=1:10
        for j=1:pattern(i).num
            % 计算待测样品与样品库样品间的余弦夹角
            d=(pattern(i).feature(:,j)'+sample')/(sqrt(sum(pattern(i).feature...
                (:',j).^2))*sqrt(sum(sample.^2)));
            % 求最大夹角及其类号
            if dmax(1)<d
                dmax(1)=d;
                dmax(2)=i-1;
            end
        end
    end
    % 输出类别
    y=dmax(2);
```

3. 效果图

运行效果如图 3-9 所示。


```

        pattern(i).feature(k,j) = 0;
    end
end
end
d = 0;% 距离
dmax = [ -inf,0];
for i = 1:10
    for j = 1:pattern(i).num
        % 计算待测样品与样品库样品间的夹角余弦
        d = (pattern(i).feature(:,j)' * sample') / (sqrt(sum(pattern(i).feature...
            (:,j).^2)) * sqrt(sum(sample.^2)));
        % 求最大夹角及其类号
        if dmax(1) < d
            dmax(1) = d;
            dmax(2) = i - 1;
        end
    end
end
% 输出类别
y = dmax(2);

```

3. 效果图

运行效果如图 3-10 所示。



图 3-10 使用二值化的夹角余弦距离法

3.2.7 二值化的 Tanimoto 测度分类

1. 实现步骤

- ① 将样本库中的每个样本进行二值化,取阈值 $T = 0.05$ 。
- ② 待测样品 X 与训练集里每个样品 X_i 的距离

$$S(\mathbf{X}, \mathbf{X}_i) = \frac{\mathbf{X}^T \mathbf{X}_i}{\mathbf{X}^T \mathbf{X} + \mathbf{X}_i^T \mathbf{X}_i - \mathbf{X}^T \mathbf{X}_i}$$

③ 循环计算待测样品和训练集中各已知样品之间的距离,找出距离待测样品最近的已知样品,该已知样品的类别就是待测样品的类别。

2. 编程代码

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% 函数名称: erzhitanimoto()
% 参数: sample: 待识别样品特征
% 返回值: y: 待识别样品所属类别
% 函数功能: 按照二值化 Tanimoto 距离法计算待测样品与样品库中的样品相似度
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
function y = erzhitanimoto( sample );

    clc;

    load templet pattern;
    % 对样品进行二值化
    for i = 1:25
        if sample(i) > 0.05
            sample(i) = 1;
        else
            sample(i) = 0;
        end
    end
    for i = 1:10
        for j = 1:pattern(i). num
            for k = 1:25
                if pattern(i). feature(k,j) > 0.05
                    pattern(i). feature(k,j) = 1;
                else
                    pattern(i). feature(k,j) = 0;
                end
            end
        end
    end
    d = 0; % 距离
    dmax = [ -inf, 0 ];
    for i = 1:10
        for j = 1:pattern(i). num
            % 计算待测样品与样品库样品间的 Tanimoto 距离
            d = (pattern(i). feature(:,j)' * sample') / (pattern(i). feature(:,j)' * ...
                pattern(i). feature(:,j) + sample * sample' - pattern(i). feature(:,j)' * ...
                * sample');
        end
    end
end
```

```
% 求最大距离及其类号
if dmax(1) < d
    dmax(1) = d;
    dmax(2) = i - 1;
end
end
end
% 输出类别
y = dmax(2);
```

3. 效果图

运行效果如图 3-11 所示。



(a) 待测样品



(b) 分类结果

图 3-11 使用二值化 Tanimoto 距离测度

本章小结

本章着重介绍了样品与样品、类与类、样品与类之间的相似性测度方法,介绍了欧式距离法、基于类中心的欧式距离法、马氏距离法、夹角余弦距离法、二值化的夹角余弦距离法和二值化的 Tanimoto 测度法等各种距离法的原理与实现步骤。采用距离法分类是本书所介绍的各种分类方法中最简单的一种,分类效果也非常理想。

习题 3

1. 简述模板匹配距离法的基本原理。
2. 简述马氏距离、欧式距离、夹角余弦距离、Tanimoto 测度的相同点与不同点。
3. 用模板匹配法编程实现英文字符的识别。

第 4 章 基于概率统计的贝叶斯分类器设计

本章要点:

- ☑ 贝叶斯决策的基本概念
- ☑ 基于最小错误率的贝叶斯决策
- ☑ 基于最小风险的贝叶斯决策
- ☑ 贝叶斯决策比较
- ☑ 基于二值数据的贝叶斯分类实现
- ☑ 基于最小错误率的贝叶斯分类实现
- ☑ 基于最小风险的贝叶斯分类实现

4.1 贝叶斯决策的基本概念

4.1.1 贝叶斯决策所讨论的问题

当分类器的设计完成后,对待测样品进行分类,一定能正确分类吗?如果有错分类情况发生,是在哪种情况下出现的?错分类的可能性有多大?这些是模式识别中所涉及的重要问题,本节用概率论的方法分析造成错分类的原因,并说明与哪些因素有关。

这里以某制药厂生产的药品检验识别为例,以此说明贝叶斯决策所要解决的问题。如图 4-1 所示,正常药品“+”,异常药品“-”。识别的目的是要依据 \mathbf{X} 向量将药品划分为两类。对于图 4-1 来说,可以用一直线作为分界线,这条直线是关于 \mathbf{X} 的线性方程,称为线性分类器。如果 \mathbf{X} 向量被划分到直线右侧,则其为正常药品,若被划分到直线左侧,则其为异常药品,可见对其作出决策是很容易的,也不会出现什么差错。

问题在于可能会出现模棱两可的情况,如图 4-2 所示。此时,任何决策都存在判错的可能性。由图 4-2 可见,在直线 A 、 B 之间,属于不同类的样品在特征空间中相互穿插,很难用简单的分界线将它们完全分开,即所观察到的某一样品的特征向量为 \mathbf{X} ,在 M 类中又有不止一类可能呈现这一 \mathbf{X} 值,无论直线参数如何设计,总会有错分类发生。如果以错分类最小为原则分类,则图中 A 直线可能是最佳的分界线,它使错分类的样品数量为最小。但是如果将一个“-”样品错分成“+”类,所造成的损失要比将“+”分成“-”类严重,这是由于将异常药品误判为正常药品,则会使病人因失去及早治疗的机会而遭受极大的损失;把正常药品误判为异常药品会给企业带来一点损失,则偏向使对“-”类样品的错分类进一步减少,可以使总的损失为最小,那么 B 直线就可能比 A 直线更适合作为分界线。可见,分类器参数的选择或者学习过程得到的结果取决于设计者选择什么样的准则函数。不同准则函数的最优解对应不同的学习结果,得到性能不同的分类器。

错分类往往难以避免,这种可能性可用 $P(\omega_i | \mathbf{X})$ 表示。如何做出合理的判决就是贝叶斯

决策所要讨论的问题。其中最有代表性的是基于最小错误率的贝叶斯决策与基于最小风险的贝叶斯决策。

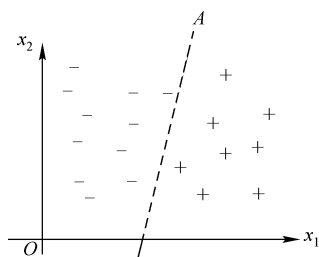


图 4-1 线性可分示意图

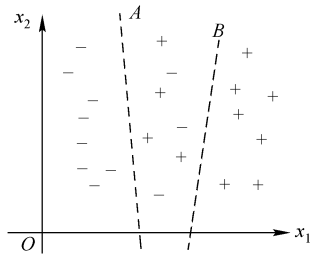


图 4-2 线性不可分示意图

(1) 基于最小错误率的贝叶斯决策

它指出机器自动识别出现错分类的条件,错分类的可能性如何计算,如何实现使错分类出现可能性最小。

(2) 基于最小错误风险的贝叶斯决策

错分类有不同情况,从图 4-2 可见,两种错误造成的损失不一样,不同的错误分类造成的损失会不相同,后一种错误更可怕,因此就要考虑减小因错分类造成的危害损失。为此,引入一种“风险”与“损失”的概念,希望做到使风险最小,减小危害大的错分类情况。

4.1.2 贝叶斯公式

若已知总共有 M 类物体,以及各类在这 n 维特征空间的统计分布,具体来说已知各类别 $\omega_i, i=1,2,\dots,M$ 的先验概率 $P(\omega_i)$ 及类条件概率密度函数 $P(\mathbf{X}|\omega_i)$ 。对于待测样品,贝叶斯公式可以计算出该样品分属各类别的概率,叫做后验概率;看 \mathbf{X} 属于那个类的可能性最大,就把 \mathbf{X} 归于可能性最大的那个类,后验概率作为识别对象归属的依据。贝叶斯公式为

$$P(\omega_i|\mathbf{X}) = \frac{P(\mathbf{X}|\omega_i)P(\omega_i)}{\sum_{j=1}^M P(\mathbf{X}|\omega_j)P(\omega_j)} \quad (4-1)$$

类别的状态是一个随机变量,而某种状态出现的概率是可以估计的。贝叶斯公式体现了先验概率、类条件概率密度函数、后验概率三者关系的式子。

1. 先验概率 $P(\omega_i)$

先验概率 $P(\omega_i)$ 针对 M 个事件出现的可能性而言,不考虑其他任何条件。例如,由统计资料表明总药品数为 N ,其中正常药品数为 N_1 ,异常药品数为 N_2 ,则

$$P(\omega_1) = \frac{N_1}{N} \quad (4-2)$$

$$P(\omega_2) = \frac{N_2}{N} \quad (4-3)$$

我们称 $P(\omega_1)$ 及 $P(\omega_2)$ 为先验概率。显然在一般情况下正常药品占比例大, 即 $P(\omega_1) > P(\omega_2)$ 。仅按先验概率来决策, 就会把所有药品都划归为正常药品, 并没有达到将正常药品与异常药品区分开的目的。这表明由先验概率所提供的信息太少。

2. 类条件概率密度函数 $P(X | \omega_i)$

类条件概率密度函数 $P(X | \omega_i)$ 是指在已知某类别的特征空间中, 出现特征值 X 的概率密度, 指第 ω_i 类样品其属性 X 是如何分布的。假定只用其一个特征进行分类, 即 $n = 1$, 并已知这两类的类条件概率密度函数分布, 如图 4-3 所示, 概率密度函数 $P(X | \omega_1)$ 是正常药品的属性分布, 概率密度函数 $P(X | \omega_2)$ 是异常药品的属性分布。

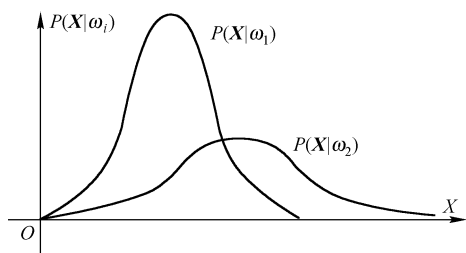


图 4-3 类条件概率

例如, 全世界华人占地球上人口总数的 20%, 但各个国家华人所占当地人口比例是不同的, 类条件概率密度函数 $P(X | \omega_i)$ 是指 ω_i 条件下出现 X 的概率密度, 在这里指第 ω_i 类样品其属性 X 是如何分布的。

在工程上的许多问题中, 统计数据往往满足正态分布规律。正态分布简单、分析方便、参量少, 是一种适宜的数学模型。如果采用正态密度函数作为类条件概率密度的函数形式, 则函数内的参数, 如期望和方差是未知的。那么问题就变成了如何利用大量样品对这些参数进行估计, 只要估计出这些参数, 类条件概率密度函数 $P(X | \omega_i)$ 也就确定了。

单变量正态密度函数为

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \quad (4-4)$$

μ 为数学期望 (均值)

$$\mu = E(x) = \int_{-\infty}^{+\infty} xP(x) dx \quad (4-5)$$

σ^2 为方差

$$\sigma^2 = E[(x - \mu)^2] = \int_{-\infty}^{+\infty} (x - \mu)^2 P(x) dx \quad (4-6)$$

多维正态密度函数为

$$P(X) = \frac{1}{(2\pi)^{n/2} |S|^{1/2}} \exp\left[-\frac{1}{2}(X - \bar{\mu})^T S^{-1} (X - \bar{\mu})\right] \quad (4-7)$$

式中, $X = (x_1, x_2, \dots, x_n)$ 为 n 维特征向量; $\bar{\mu} = (\mu_1, \mu_2, \dots, \mu_n)$ 为 n 维均值向量; $S = E[(X - \bar{\mu})(X - \bar{\mu})^T]$ 为 n 维协方差矩阵; S^{-1} 是 S 的逆矩阵; $|S|$ 是 S 的行列式。

在大多数情况下, 类条件密度可以采用多维变量的正态密度函数来模拟。

$$\begin{aligned} P(X | \omega_i) &= \frac{1}{(2\pi)^{n/2} |S_i|^{1/2}} \exp\left[-\frac{1}{2}(X - \bar{X}^{(\omega_i)})^T S_i^{-1} (X - \bar{X}^{(\omega_i)})\right] \\ &= -\frac{1}{2}(X - \bar{X}^{(\omega_i)})^T S_i^{-1} (X - \bar{X}^{(\omega_i)}) - \frac{n}{2} \ln 2\pi - \frac{1}{2} \ln |S_i| \end{aligned} \quad (4-8)$$

$\bar{X}^{(\omega_i)}$ 为 ω_i 类的均值向量。

3. 后验概率

后验概率是指呈现状态 \mathbf{X} 时,该样品分属各类别的概率,这个概率值可以作为识别对象归属的依据。由于属于不同类的待识别对象存在着呈现相同观测值的可能,即所观测到的某一样品的特征向量为 \mathbf{X} ,而在类中又有不止一类可能呈现这一 \mathbf{X} 值,它属于各类的概率又是多少呢?这种可能性可用 $P(\omega_i | \mathbf{X})$ 表示。可以利用贝叶斯公式来计算这种条件概率,称为状态的后验概率 $P(\omega_i | \mathbf{X})$ 。

$$P(\omega_i | \mathbf{X}) = \frac{P(\mathbf{X} | \omega_i)P(\omega_i)}{\sum_{j=1}^M P(\mathbf{X} | \omega_j)P(\omega_j)} \quad (4-9)$$

$P(\omega_i | \mathbf{X})$ 表示在 \mathbf{X} 出现的条件下,样品为 ω_i 类的概率。在这里要弄清楚条件概率这个概念。 $P(A | B)$ 是条件概率的通用符号,在“|”后边出现 B 的为条件,之前的 A 为某个事件,即在某条件 B 下出现某个事件 A 的概率。

4. $P(\omega_1 | \mathbf{X})$ 和 $P(\omega_2 | \mathbf{X})$ 与 $P(\mathbf{X} | \omega_1)$ 和 $P(\mathbf{X} | \omega_2)$ 的区别

① $P(\omega_1 | \mathbf{X})$ 和 $P(\omega_2 | \mathbf{X})$ 是在同一条件 \mathbf{X} 下,比较 ω_1 与 ω_2 出现的概率,若 $P(\omega_1 | \mathbf{X}) >$

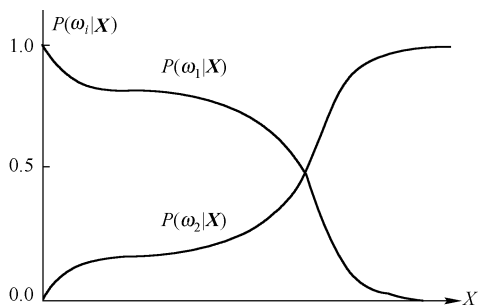


图 4-4 后验概率比较图

$P(\omega_2 | \mathbf{X})$, 则可以下结论,在 \mathbf{X} 条件下,事件 ω_1 出现的可能性大,如图 4-4 所示。两类情况下,则有 $P(\omega_1 | \mathbf{X}) + P(\omega_2 | \mathbf{X}) = 1$ 。

② $P(\mathbf{X} | \omega_1)$ 与 $P(\mathbf{X} | \omega_2)$ 都是指各自条件下出现 \mathbf{X} 的可能性,两者之间没有联系,比较两者没有意义。 $P(\mathbf{X} | \omega_1)$ 和 $P(\mathbf{X} | \omega_2)$ 是在不同条件下讨论的问题,即使只有两类 ω_1 与 ω_2 , $P(\mathbf{X} | \omega_1) + P(\mathbf{X} | \omega_2) \neq 1$ 。不能仅因为 $P(\mathbf{X} | \omega_1) > P(\mathbf{X} | \omega_2)$, 就认为 \mathbf{X} 是第一类事物的可能性较大。只

有考虑先验概率这一因素,才能决定 \mathbf{X} 条件下,判为 ω_1 类或 ω_2 类的可能性比较大。

4.2 基于最小错误率的贝叶斯决策

假定得到一个待识别量的特征 \mathbf{X} 后,每个样品 \mathbf{X} 有 n 个特征,即 $\mathbf{X} = (x_1, x_2, \dots, x_n)^T$,通过样品库,计算先验概率 $P(\omega_i)$ 及类别条件概率密度函数 $P(\mathbf{X} | \omega_i)$,得到呈现状态 \mathbf{X} 时,该样品分属各类别的概率,显然这个概率值可以作为识别对象判属的依据。从后验概率分布图 4-4 可见,在 \mathbf{X} 值小时,药品被判为正常是比较合理的,判断错误的可能性小。基于最小错误概率的贝叶斯决策就是按后验概率的大小判决的。这个规则又可以根据类别数目,写成不同的几种等价形式。

1. 两类问题

若每个样品属于 ω_1, ω_2 类中的一类,已知两类的先验概率分别为 $P(\omega_1), P(\omega_2)$,两类的类条件概率密度为 $P(\mathbf{X} | \omega_1), P(\mathbf{X} | \omega_2)$ 。则任给一 \mathbf{X} ,判断 \mathbf{X} 的类别。由贝叶斯公式可知

$$P(\omega_j | \mathbf{X}) = P(\mathbf{X} | \omega_j)P(\omega_j)/P(\mathbf{X}) \quad (4-10)$$

由全概率公式可知

$$P(\mathbf{X}) = \sum_{j=1}^M P(\mathbf{X} | \omega_j)P(\omega_j) \quad (4-11)$$

其中 M 为类别。

对于两类问题

$$P(\mathbf{X}) = P(\mathbf{X} | \omega_1)P(\omega_1) + P(\mathbf{X} | \omega_2)P(\omega_2) \quad (4-12)$$

所以用后验概率来判别为

$$P(\omega_1 | \mathbf{X}) \begin{cases} > \\ < \end{cases} P(\omega_2 | \mathbf{X}) \Rightarrow \mathbf{X} \in \begin{cases} \omega_1 \\ \omega_2 \end{cases} \quad (4-13)$$

判别函数还有另外两种形式。

(1) 似然比形式

$$l(\mathbf{X}) = \frac{P(\mathbf{X} | \omega_1)}{P(\mathbf{X} | \omega_2)} \begin{cases} > \frac{P(\omega_2)}{P(\omega_1)} \\ < \frac{P(\omega_2)}{P(\omega_1)} \end{cases} \Rightarrow \mathbf{X} \in \begin{cases} \omega_1 \\ \omega_2 \end{cases} \quad (4-14)$$

其中,式(4-14)中的 $l(\mathbf{X})$ 在统计学中称为似然比,而 $\frac{P(\omega_2)}{P(\omega_1)}$ 称为似然比阈值。

(2) 对数形式

$$\ln P(\mathbf{X} | \omega_1) - \ln P(\mathbf{X} | \omega_2) \begin{cases} > \ln P(\omega_2) - P(\omega_1) \\ < \ln P(\omega_2) - P(\omega_1) \end{cases} \Rightarrow \mathbf{X} \in \begin{cases} \omega_1 \\ \omega_2 \end{cases} \quad (4-15)$$

式(4-13)、式(4-14)、式(4-15)三种判别函数是一致的,也可以用后验概率来表示判别函数。

2. 多类问题

现在讨论多类问题的情况。在第1章已经介绍了判别函数的一般形式,如图4-5所示。

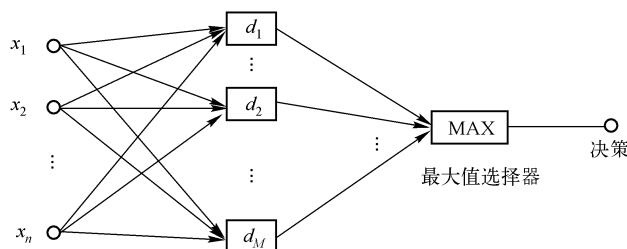


图 4-5 多类问题判别

若样本分为 M 类 $\omega_1, \omega_2, \dots, \omega_M$, 各类的先验概率分别为 $P(\omega_1), P(\omega_2), \dots, P(\omega_M)$, 各类的类条件概率密度分别为 $P(\mathbf{X} | \omega_1), P(\mathbf{X} | \omega_2), \dots, P(\mathbf{X} | \omega_M)$, 就有 M 个判别函数。在取得一个观察特征 \mathbf{X} 之后, 在特征 \mathbf{X} 的条件下, 看哪个类的概率最大, 应该把 \mathbf{X} 归于概率最大的那个类。因此对于任一模式 \mathbf{X} , 可以通过比较各个判别函数来确定 \mathbf{X} 的类别。

$$P(\omega_i)P(\mathbf{X} | \omega_i) = \max_{1 \leq j \leq M} \{P(\omega_j)P(\mathbf{X} | \omega_j)\} \Rightarrow \mathbf{X} \in \omega_i \quad i = 1, 2, \dots, M \quad (4-16)$$

就是把 \mathbf{X} 代入 M 个判别函数中, 看哪个判别函数最大, 就把 \mathbf{X} 归于这一类。

判别函数的对数形式为

$$\ln P(\omega_i) + \ln P(\mathbf{X} | \omega_i) = \max_{1 \leq j \leq M} \{ \ln P(\omega_j) + \ln P(\mathbf{X} | \omega_j) \} \Rightarrow \mathbf{X} \in \omega_i \quad i = 1, 2, \dots, M \quad (4-17)$$

由于先验概率通常是很容易求出的,贝叶斯分类器的核心问题就是求出类条件概率密度 $P(\mathbf{X} | \omega_i)$, 如果求出了条件概率, 则后验概率就可以求出了, 判别问题就解决了。在大多数情况下, 类条件密度可以采用多维变量的正态密度函数来模拟。所以此时正态分布的贝叶斯分类器判别函数为

$$\begin{aligned} h_i(\mathbf{X}) &= P(\mathbf{X} | \omega_i) P(\omega_i) = \frac{1}{(2\pi)^{n/2} |\mathbf{S}_i|^{1/2}} \exp \left[-\frac{1}{2} (\mathbf{X} - \overline{\mathbf{X}}^{(\omega_i)})^T \mathbf{S}_i^{-1} (\mathbf{X} - \overline{\mathbf{X}}^{(\omega_i)}) \right] P(\omega_i) \\ &= -\frac{1}{2} (\mathbf{X} - \overline{\mathbf{X}}^{(\omega_i)})^T \mathbf{S}_i^{-1} (\mathbf{X} - \overline{\mathbf{X}}^{(\omega_i)}) - \frac{n}{2} \ln 2\pi - \frac{1}{2} \ln |\mathbf{S}_i| + \ln P(\omega_i) \end{aligned} \quad (4-18)$$

使用什么样的决策原则可以做到错误率最小呢? 前提是要知道一个样品 \mathbf{X} 分属不同类别的可能性, 表示成 $P(\omega_i | \mathbf{X})$, 然后根据后验概率最大的类来分类。后验概率要通过贝叶斯公式从先验概率与类分布函数来计算。

3. 最小错误率证明

基于最小错误率的贝叶斯决策根据: 如果

$$P(\omega_i | \mathbf{X}) = \max_{j=1,2} P(\omega_j | \mathbf{X}), \text{ 则 } \mathbf{X} \in \omega_i \quad (4-19)$$

由于统计判别方法是基于统计参数做出决策的, 因此错误率也只能从平均意义上讲, 表示为在观测值可能取值的整个范围内错误率的均值。

为了直观, 假设 \mathbf{X} 只有一个特征, $n=1$, 于是 $P(\mathbf{X} | \omega_1)$, $P(\mathbf{X} | \omega_2)$ 都是一元函数, 将整个特征空间分为不相交的两个部分 R_1 和 R_2 。当模式落在 R_1 内判定它属于 ω_1 类, 求分类器相当于求 R_1 和 R_2 的分界线。

(1) 第一类判错

如果 \mathbf{X} 原属于 ω_1 类, 却落在 R_2 内, 称为第一类判错, 错误率为

$$P_1(e) = P(\mathbf{X} \in R_2 | \omega_1) = \int_{R_2} P(\mathbf{X} | \omega_1) d\mathbf{x}$$

(2) 第二类判错

如果 \mathbf{X} 原属于 ω_2 类, 却落在 R_1 内, 称为第二类判错, 错误率为

$$P_2(e) = P(\mathbf{X} \in R_1 | \omega_2) = \int_{R_1} P(\mathbf{X} | \omega_2) d\mathbf{x}$$

因此, 平均错误率 $P(e)$ 可表示成

$$P(e) = \int_{R_2} P(\mathbf{X} | \omega_1) p(\omega_1) d\mathbf{X} + \int_{R_1} P(\mathbf{X} | \omega_2) p(\omega_2) d\mathbf{X} \quad (4-20)$$

因此, 错误率为图中两个画线部分之和, 如图 4-6 所示。

贝叶斯决策式(4-19)表明每个样品所属类别都使 $P(\omega_i | \mathbf{X})$ 为最大, 实际上使 \mathbf{X} 判错的可能性达到最小, 这时总的错误率为最小。按贝叶斯决策分类时, $\int_{R_2} P(\mathbf{X} | \omega_1) p(\omega_1) d\mathbf{X} = \int_{R_1} P(\mathbf{X} | \omega_2) p(\omega_2) d\mathbf{X}$ 。

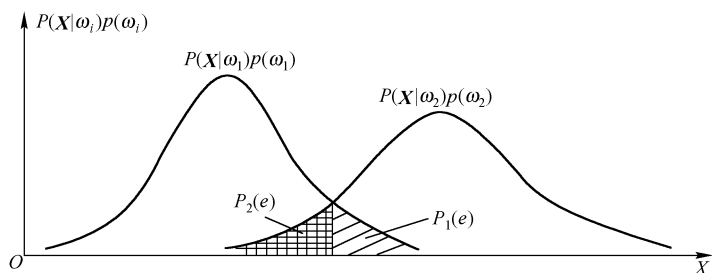


图 4-6 贝叶斯平均错误率最小示意图

4.3 基于最小风险的贝叶斯决策

上面讨论了使错误率最小的贝叶斯决策规则。然而,当接触到实际问题时,可以发现使错误率最小并不一定是一个普遍适用的最佳选择,如图 4-7 所示。

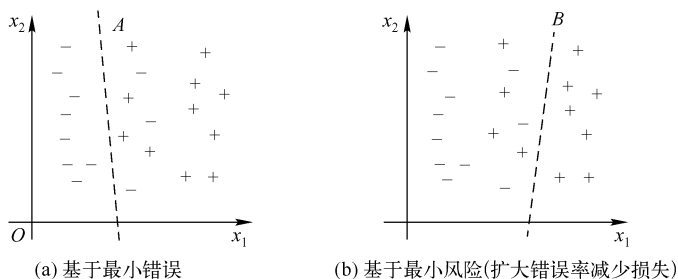


图 4-7 基于最小错误分类和基于最小风险分类比较

直线 B 的划分把正常药品误判为异常药品,这样会扩大总错误率,给企业带来一些损失;直线 A 的划分将异常药品误判为正常药品,虽然使错误分类最小,但会使患者因失去正确的治疗而遭受极大的损失。可见使错误率最小并不一定是最佳选择。

实际应用时,从根据不同性质的错误会引起不同程度的损失考虑出发,宁肯扩大一些总的错误率,但也要使总的损失减少。这时直线 B 的划分为最实用。这会引进一个与损失有关的概念——风险。在做出决策时,要考虑所承担的风险。基于最小风险的贝叶斯决策规则正是为了体现这一点而产生的。

基于最小错误概率,在分类时取决于观测值 \mathbf{X} 对各类的后验概率中的最大值,因而也就无法估计做出错误决策所带来的损失。为此,不妨将做出判决的依据,从单纯考虑后验概率最大值,改为对该观测值 \mathbf{X} 条件下各状态后验概率求加权的方式,表示成

$$R_i(\mathbf{X}) = \sum_{j=1}^M \lambda(\alpha_i, j) P(\omega_j | \mathbf{X}) \quad (4-21)$$

式中, α_i 代表将 \mathbf{X} 判为 ω_i 类的决策; $\lambda(\alpha_i, j)$ 表示观测样品 \mathbf{X} 实属于 ω_j , 由于采用 α_i 决策而被判为 ω_i 时所造成的损失; R_i 则表示观测值 \mathbf{X} 被判为 i 类时损失的均值; $\lambda(\alpha_1, 2)$ 表示 \mathbf{X} 确实是异常药品(ω_2), 但采取决策 α_1 被判定为正常(ω_1), 则会有损失 $\lambda(\alpha_1, 2)$; $\lambda(\alpha_2, 1)$ 表示 \mathbf{X} 确实是正常(ω_1), 却采取决策 α_2 被判定为异常(ω_2), 则会损失 $\lambda(\alpha_2, 1)$ 。

损失函数 $\lambda(\alpha_1, 2)$ 比 $\lambda(\alpha_2, 1)$ 大, 另外为了使式子写得更方便, 也可以定义 $\lambda(\alpha_1, 1)$ 与

$\lambda(\alpha_2, 2)$, 是指正确判断也可有损失。那么把 X 判做 ω_1 引进的损失应该与 $\lambda(\alpha_1, 2)$ 以及 $\lambda(\alpha_2, 1)$ 都有关, 哪一个占主要成分, 则取决于 $P(\omega_1 | X)$ 与 $P(\omega_2 | X)$, 因此变成了一个加权和, 如表 4-1 所示。

表 4-1 风险分析

X	采取 α_1 决策, 将 X 判为正常 ω_1 的风险 $R_1(X)$	采取 α_2 决策, 将 X 判为异常 ω_2 的风险 $R_2(X)$
X 为正常	损失: $\lambda(\alpha_1, 1)$ 风险: $\lambda(\alpha_1, 1)P(\omega_1 X)$	损失: $\lambda(\alpha_2, 1)$ 正常(ω_1)被判为异常(ω_2) 风险: $\lambda(\alpha_2, 1)P(\omega_1 X)$
X 为异常	损失: $\lambda(\alpha_1, 2)$ 异常(ω_2)被判做正常(ω_1) 风险: $\lambda(\alpha_1, 2)P(\omega_2 X)$	损失: $\lambda(\alpha_2, 2)$ 风险: $\lambda(\alpha_2, 2)P(\omega_2 X)$
总风险	$R_1(X) = \lambda(\alpha_1, 1)P(\omega_1 X) + \lambda(\alpha_1, 2)P(\omega_2 X)$	$R_2(X) = \lambda(\alpha_2, 1)P(\omega_1 X) + \lambda(\alpha_2, 2)P(\omega_2 X)$

此时做出哪一种决策就要看 $R_1(X)$ 小还是 $R_2(X)$ 小了, 这就是基于最小风险的贝叶斯决策的基本出发点。如果希望尽可能避免将某状态 ω_j 错判为状态 ω_i , 则可将相应的 $\lambda(\alpha_i, j)$ 值选择得大些, 以表明损失的严重性。加权和 R_i 用来衡量观测样品 X 被判为状态 ω_i 所需承担的风险。而究竟将 X 判为哪类则应依据所有 $R_i (i = 1, \dots, M)$ 中的最小值, 即最小风险来确定。一般 $\lambda(\alpha_1, 1) = \lambda(\alpha_2, 2) = 0$, 为了避免将异常药品判为正常的严重损失, 取 $\lambda(\alpha_1, 2) > \lambda(\alpha_2, 1)$ 则会使 $R_2(X) < R_1(X)$ 机会多, 根据贝叶斯最小风险分类法, 表明正常药品错判为异常的可能性大于异常药品错判为正常的可能性, 损失减小。

(1) 贝叶斯决策的相关定义

① 自然状态与状态空间。其中自然状态是指待识别对象的类别, 而状态空间 Ω 则是由所有自然状态组成的空间, $\Omega = \{\omega_1, \omega_2, \dots, \omega_M\}$ 。

② 决策与决策空间。在决策论中, 对分类问题所做的判决, 称之为决策, 由所有决策组成的空间称为决策空间。决策不仅包括根据观测值将样品归到哪一类, 还可包括其他决策, 如“拒绝”等。在不考虑“拒绝”情况下, 决策空间内决策总数等于类别数 M , 表示成: $A = \{\alpha_1, \alpha_2, \dots, \alpha_M\}$ 。

③ 损失函数 $\lambda(\alpha_i, j)$ 。它明确表示本身属于自然状态 ω_j , 做出决策 α_i , 使其归属于 ω_i 所造成的损失。

④ 观测值 X 条件下的期望损失 $R(\alpha_i | X)$, R_i 也称为条件风险。

$$R(\alpha_i | X) = \sum_{j=1}^M \lambda(\alpha_i, j)P(\omega_j | X) \quad i = 1, 2, \dots, M \quad (4-22)$$

⑤ 最小风险贝叶斯决策规则可写成

$$R(\alpha_k | X) = \min_{i=1, \dots, M} R(\alpha_i | X) \quad (4-23)$$

这里计算的是最小值。

(2) 最小风险贝叶斯决策的操作步骤

① 已知 $P(\omega_i), P(X | \omega_i), i = 1, \dots, M$ 及给出待识别 X 的情况下, 根据贝叶斯公式计算出

后验概率

$$P(\omega_i | X) = \frac{P(X | \omega_i)P(\omega_i)}{\sum_{j=1}^M P(X | \omega_j)P(\omega_j)} \quad j = 1, \dots, M$$

② 利用计算出的后验概率及决策表,按式(4-22)计算出采取决策 $\alpha_i, i = 1, \dots, M$ 的条件风险

$$R(\alpha_i | X) = \sum_{j=1}^M \lambda(\alpha_i, j)P(\omega_j | X) \quad i = 1, 2, \dots, M$$

③ 对②中得到的 M 个条件风险值 $R(\alpha_i | X), i = 1, \dots, M$ 进行比较,找出使条件风险最小的决策 α_k ,则 α_k 就是最小风险贝叶斯决策, ω_k 就是待识别样品 X 的归类。

4.4 贝叶斯决策比较

1. 最小错误率与最小风险的贝叶斯决策比较

讨论一下最小错误率与最小风险的贝叶斯决策之间的关系,设损失函数为

$$\lambda(\alpha_i, \omega_j) = \begin{cases} 0, & i=j \\ 1, & i \neq j \end{cases} \quad i, j = 1, 2, \dots, M \quad (4-24)$$

式中,假定对 M 类只有 M 个决策,即不考虑“拒绝”等其他情况,式(4-24)表明,当做出正确决策(即 $i=j$)时没有损失,而对于任何错误决策,其损失均为 1。这样定义的损失函数称为 0-1 损失函数,如表 4-2 所示。

表 4-2 最小错误率与最小风险的贝叶斯决策之间的关系

X	采取 α_1 决策将 X 判为正常 ω_1 的风险 $R_1(X)$	采取 α_2 决策将 X 判为异常 ω_2 的风险 $R_2(X)$
X 为正常	损失: $\lambda(\alpha_1, 1) = 0$ 风险: $\lambda(\alpha_1, 1)P(\omega_1 X) = 0$	损失: $\lambda(\alpha_2, 1) = 1$ 正常(ω_1)被判定为异常(ω_2) 风险: $\lambda(\alpha_2, 1)P(\omega_1 X) = P(\omega_1 X)$
X 为异常	损失: $\lambda(\alpha_1, 2) = 1$ 异常(ω_2)被判为正常(ω_1) 风险: $\lambda(\alpha_1, 2)P(\omega_2 X) = P(\omega_2 X)$	损失: $\lambda(\alpha_2, 2) = 0$ 风险: $\lambda(\alpha_2, 2)P(\omega_2 X) = 0$
总风险	$R_1(X) = \lambda(\alpha_1, 1)P(\omega_1 X) +$ $\lambda(\alpha_1, 2)P(\omega_2 X) = P(\omega_2 X)$ $P(\omega_2 X)$ 是将 X 判为 ω_2 时的错误概率	$R_2(X) = \lambda(\alpha_2, 1)P(\omega_1 X) + \lambda(\alpha_2, 2)$ $P(\omega_2 X) = P(\omega_1 X)$ $P(\omega_1 X)$ 是将 X 判为 ω_1 时的错误概率

由表 4-2 可见,当 $P(\omega_2 | X) > P(\omega_1 | X)$ 时,基于最小错误概率的贝叶斯决策结果是 ω_2 类;而此时 $R_2(X) = P(\omega_1 | X), R_1(X) = P(\omega_2 | X), R_2(X) < R_1(X)$ 基于最小风险的贝叶斯决策结果同样也是 ω_2 类。因此,在 0-1 损失函数情况下,基于最小风险的贝叶斯决策结果,也就是基于最小错误概率的贝叶斯决策结果。

实际上, $\sum_{j=1, j \neq i}^M P(\omega_j | \mathbf{X})$ 也是将 \mathbf{X} 判为 ω_i 时的错误概率, $\sum_{j=1, j \neq i}^M P(\omega_j | \mathbf{X}) = 1 - P(\omega_i | \mathbf{X})$, 因此, 当 $P(\omega_i | \mathbf{X})$ 最大时, 基于最小错误概率的贝叶斯决策结果, 将该样品判归为 ω_i 类, 而此时 $R_i(\mathbf{X})$ 最小, 风险也是最小的, 它与基于最小错误率的贝叶斯决策的判据是一样的。

2. 实例比较

某制药厂生产产品检测分两种情况: 正常 (ω_1) 和异常 (ω_2), 两类的先验概率分别为 $P(\omega_1) = 0.95, P(\omega_2) = 0.05$ 。现有一待测产品呈现出状态 \mathbf{X} , 由类条件概率密度分布曲线求得 $P(\mathbf{X} | \omega_1) = 0.3, P(\mathbf{X} | \omega_2) = 0.5$, 试对该产品 \mathbf{X} 按最小错误率的贝叶斯决策进行分类。若在上述条件基础之上, 已知 $\lambda_{11} = 0, \lambda_{12} = 15, \lambda_{21} = 1, \lambda_{22} = 0$ (λ_{11} 表示 $\lambda(\alpha_1, \omega_1)$ 的简写), 按最小风险贝叶斯决策进行分类。对这两种分类结果进行比较, 如表 4-3 所示。

表 4-3 两种贝叶斯决策分类结果比较

基于最小错误率分类	基于最小风险分类
<p>解: 利用贝叶斯公式, 分别计算出状态为 \mathbf{X} 时 ω_1 与 ω_2 的后验概率</p> $P(\omega_1 \mathbf{X}) = \frac{P(\mathbf{X} \omega_1)P(\omega_1)}{\sum_{j=1}^2 P(\mathbf{X} \omega_j)P(\omega_j)}$ $= \frac{0.3 \times 0.95}{0.3 \times 0.95 + 0.5 \times 0.05} = 0.919$ <p>而 $P(\omega_2 \mathbf{X}) = 1 - P(\omega_1 \mathbf{X}) = 0.081$</p> <p>根据贝叶斯决策则有</p> $P(\omega_1 \mathbf{X}) = 0.919 > P(\omega_2 \mathbf{X}) = 0.081$ <p>状态 \mathbf{X} 属于类别 ω_1 的可能性远比属于类别 ω_2 的可能性大, 将该产品判为正常产品比较合理</p>	<p>解: 已知条件为</p> $P(\omega_1) = 0.95, P(\omega_2) = 0.05$ $P(\mathbf{X} \omega_1) = 0.3, P(\mathbf{X} \omega_2) = 0.5$ $\lambda_{12} = 0, \lambda_{11} = 15, \lambda_{21} = 1, \lambda_{22} = 0$ <p>可知后验概率为</p> $P(\omega_1 \mathbf{X}) = 0.919, P(\omega_2 \mathbf{X}) = 0.081, \text{再按式(4-22)计算出条件风险}$ $R(\alpha_1 \mathbf{X}) = \sum_{j=1}^2 \lambda_{1j} P(\omega_j \mathbf{X}) = \lambda_{12} P(\omega_2 \mathbf{X}) = 1.215$ $R(\alpha_2 \mathbf{X}) = \sum_{j=1}^2 \lambda_{2j} P(\omega_j \mathbf{X}) = \lambda_{21} P(\omega_1 \mathbf{X}) = 0.919$ <p>由于 $R(\alpha_1 \mathbf{X}) > R(\alpha_2 \mathbf{X})$, 即决策为 ω_2 的条件风险小于决策为 ω_1 的条件风险, 因此应采取决策行动 α_2, 即判待识别的产品 \mathbf{X} 为 ω_2 类——异常产品</p>

通过两例比较, 其分类结果正好相反, 这是因为影响决策结果的因素又多了一个“损失”。由于两类错误决策所造成的损失相差很悬殊, 因此“损失”在这里起了主导作用。

从上述讨论可以看出, 正确制定损失函数值, 是基于最小风险的贝叶斯决策方法在实际应用中的一个关键问题。在实际中列出合适的决策表并不是一件容易的事, 需根据所研究的具体问题, 分析错误决策造成损失的严重程度。

4.5 基于二值数据的贝叶斯分类实现

1. 理论基础

所谓二值数据, 即各样品的每一特征只取数值 1 或 0。对于手写数字的分类问题, 在每个

图 4-8 手写数字的二值化示意图

① 计算先验概率 $P(\omega_i)$, 先验概率可由各类的样品数和样品总数近似计算:

$$P(\omega_i) \approx N_i/N \quad i=0,1,2,\cdots,9$$

② 计算 $P_i(\omega_i)$, 再计算类条件概率 $P(X | \omega_i)$:

$$P_j(\omega_i) = \left(\sum_{\substack{k=0 \\ X \in \omega:}}^{N_i} x_{kj} + 1 \right) \mid (N_i + 2) \quad i = 0, 1, 2, \dots, 9 \text{ 共 } 10 \text{ 个类};$$

$j = 0, 1, 2, \dots, 24$, 共 25 个特征

$$P(x_i = 1 \mid \mathbf{X} \in \omega_i) = P_i(\omega_i)$$

$$P(x_i=0 \mid X \in \omega_i) = 1 - P_i(\omega_i) \quad i=0,1,2,\dots,9; j=0,1,2,\dots,24$$

$$P(X|\omega_i) = P[X = (x_0, x_1, x_2, \dots, x_{24}) | X \in \omega_i]$$

$$= \prod_{i=0}^{24} P(x_j = \alpha \mid \mathbf{X} \in \omega_i) \quad i = 0, 1, 2, \dots, 9$$

③ 应用贝叶斯公式求后验概率

$$P(\omega_i | X) = \frac{P(\omega_i)P(X | \omega_i)}{P(\omega_0)P(X | \omega_0) + P(\omega_1)P(X | \omega_1) + \dots + P(\omega_9)P(X | \omega_9)} \quad i=0,1,2,\dots,9$$

3. 编程代码

```
% 函数名称:baveserzhi( )
```

```

% 参数:sample:待识别样品特征
% 返回值: y:待识别样品所属类别
% 函数功能:基于二值数据的贝叶斯分类器
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = bayeserzhi( sample )
clc;
load templet pattern
sum = 0;
pw = [ ];% 先验概率
p = [ ];
pxw = [ ];% 类条件概率
pwx = 0;% 后验概率
% 求先验概率
for i = 1:10
    sum = sum + pattern(i). num;
end
for i = 1:10
    pw(i) = pattern(i). num/sum;
end
% 求类条件概率
for i = 1:10
    for j = 1:25
        sum = 0;
        for k = 1:pattern(i). num
            if( pattern(i). feature(j,k) > 0. 05)
                sum = sum + 1;
            end
        end
        p(j,i) = ( sum + 1)/( pattern(i). num + 2);
    end
end
for i = 1:10
    sum = 1;
    for j = 1:25
        if( sample(j) > 0. 05)
            sum = sum * p(j,i);
        else
            sum = sum * ( 1 - p(j,i) );
        end
    end
    pxw(i) = sum;
end
% 求后验概率

```

```

sum = 0;
for i = 1:10
    sum = sum + pw(i) * pxw(i);
end
for i = 1:10
    pxw(i) = pw(i) * pxw(i)/sum;
end
[ maxval maxpos ] = max( pxw );
% 返回类别号
y = maxpos - 1;

```

4. 效果图

基于二值数据的贝叶斯分类实现效果如图 4-9 所示。



图 4-9 基于二值数据的贝叶斯分类实现效果

4.6 基于最小错误率的贝叶斯分类实现

1. 理论总结

错误率最小的贝叶斯分类器设计思想是寻找一种划分方式,使“错判”率最小。

(1) 两类问题

若两类样品都满足正态分布,最小错误概率的贝叶斯分类器可化为

$$h(\mathbf{X}) = \frac{1}{2}(\mathbf{X} - \overline{\mathbf{X}}^{(\omega_1)})^T \mathbf{S}_1^{-1}(\mathbf{X} - \overline{\mathbf{X}}^{(\omega_1)}) - \frac{1}{2}(\mathbf{X} - \overline{\mathbf{X}}^{(\omega_2)})^T \mathbf{S}_2^{-1}(\mathbf{X} - \overline{\mathbf{X}}^{(\omega_2)}) + \frac{1}{2} \ln \frac{|\mathbf{S}_1|}{|\mathbf{S}_2|} - \ln \frac{P(\omega_1)}{P(\omega_2)} \begin{cases} < 0 \\ > 0 \end{cases} \Rightarrow \mathbf{X} \in \begin{cases} \omega_1 \\ \omega_2 \end{cases}$$

若两类样品不仅呈正态分布,而且协方差矩阵相等,即 $\mathbf{S}_1 = \mathbf{S}_2 = \mathbf{S}$,则贝叶斯分类器可进一步简化为

$$h(\mathbf{X}) = (\overline{\mathbf{X}}^{(\omega_2)} - \overline{\mathbf{X}}^{(\omega_1)})^T \mathbf{S}^{-1} \mathbf{X} + \frac{1}{2}(\overline{\mathbf{X}}^{(\omega_1)^T} \mathbf{S}^{-1} \overline{\mathbf{X}}^{(\omega_1)} - \overline{\mathbf{X}}^{(\omega_2)^T} \mathbf{S}^{-1} \overline{\mathbf{X}}^{(\omega_2)}) - \ln \frac{P(\omega_1)}{P(\omega_2)} \begin{cases} < 0 \\ > 0 \end{cases} \Rightarrow \mathbf{X} \in \begin{cases} \omega_1 \\ \omega_2 \end{cases}$$

(2) 多类问题

当分类数 $M > 2$ 时,比较 $P(\omega_1)P(\mathbf{X}|\omega_1), P(\omega_2)P(\mathbf{X}|\omega_2), \dots, P(\omega_M)P(\mathbf{X}|\omega_M)$ 的大小,并且有

$$P(\omega_i)P(\mathbf{X}|\omega_i) = \max_{1 \leq l \leq M} \{P(\omega_l)P(\mathbf{X}|\omega_l)\} \Rightarrow \mathbf{X} \in \omega_i$$

即将 \mathbf{X} 划为 M 个函数中最大的那一类。

若 $\omega_1, \omega_2, \dots, \omega_M$ 均服从正态分布,则判别函数可以写为

$$\begin{aligned} h_i(\mathbf{X}) &= -\frac{1}{2}(\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}})^T S_i^{-1}(\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}}) + \ln P(\omega_i) - \frac{1}{2} \ln |S_i| \\ &= \max_{1 \leq l \leq M} \left\{ -\frac{1}{2}(\mathbf{X} - \overline{\mathbf{X}^{(\omega_l)}})^T S_l^{-1}(\mathbf{X} - \overline{\mathbf{X}^{(\omega_l)}}) + \ln P(\omega_l) - \frac{1}{2} \ln |S_l| \right\} \Rightarrow \mathbf{X} \in \omega_i \end{aligned}$$

若 $\omega_1, \omega_2, \dots, \omega_M$ 不仅服从正态分布,而且协方差矩阵相等,即 $S_1 = S_2 = \dots = S_M = S$,则判别函数可变为

$$h_i(\mathbf{X}) = -\frac{1}{2}(\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}})^T S^{-1}(\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}}) + \ln P(\omega_i) - \frac{1}{2} \ln |S|$$

对于每一个 $h_i(\mathbf{X})$,最后一项 $-\frac{1}{2} \ln |S|$ 都相等,可以不计。则 $h_i(\mathbf{X})$ 可以变为

$$h_i(\mathbf{X}) = -\frac{1}{2}(\mathbf{X}^T S^{-1} \mathbf{X} - \mathbf{X}^T S^{-1} \overline{\mathbf{X}^{(\omega_i)}} - \overline{\mathbf{X}^{(\omega_i)}}^T S^{-1} \mathbf{X} + \overline{\mathbf{X}^{(\omega_i)}}^T S^{-1} \overline{\mathbf{X}^{(\omega_i)}}) + \ln P(\omega_i)$$

括号内的第一项对每一个 $h_i(\mathbf{X})$ 都相同,与分类无关,可以省略。又因为

$$\overline{\mathbf{X}^{(\omega_i)}}^T S^{-1} \mathbf{X} = (\mathbf{X}^T S^{-1} \overline{\mathbf{X}^{(\omega_i)}})^T$$

是一个数值,所以括号内的第2、3项相等,可以合并。因此 $h_i(\mathbf{X})$ 可以简化为

$$h_i(\mathbf{X}) = \mathbf{X}^T S^{-1} \overline{\mathbf{X}^{(\omega_i)}} - \frac{1}{2} \overline{\mathbf{X}^{(\omega_i)}}^T S^{-1} \overline{\mathbf{X}^{(\omega_i)}} + \ln P(\omega_i)$$

或

$$\begin{aligned} \mathbf{X}^T S^{-1} \overline{\mathbf{X}^{(\omega_i)}} - \frac{1}{2} \overline{\mathbf{X}^{(\omega_i)}}^T S^{-1} \overline{\mathbf{X}^{(\omega_i)}} + \ln P(\omega_i) &= \max_{1 \leq l \leq M} \left[\mathbf{X}^T S^{-1} \overline{\mathbf{X}^{(\omega_l)}} - \frac{1}{2} \overline{\mathbf{X}^{(\omega_l)}}^T S^{-1} \overline{\mathbf{X}^{(\omega_l)}} + \ln P(\omega_l) \right] \\ &\Rightarrow \mathbf{X} \in \omega_i \end{aligned}$$

2. 实现步骤

在手写数字的识别属于多类情况,每类样品呈正态分布。为使样品协方差矩阵为正定,事先将样品库和待测样品进行主成分分析。

① 求出每一类手写数字样品的均值:

$$\overline{\mathbf{X}^{(\omega_i)}} = \frac{1}{N_i} \sum_{\mathbf{X} \in \omega_i} \mathbf{X} = (\overline{x_1^{(\omega_i)}}, \overline{x_2^{(\omega_i)}}, \dots, \overline{x_n^{(\omega_i)}})^T \quad i = 0, 1, 2, \dots, 9$$

式中, N_i 代表 ω_i 类的样品个数; n 代表特征数目。

② 求每一类的协方差矩阵:

$$s_{jk}^i = \frac{1}{N_i - 1} \sum_{l=1}^{N_i} (x_{lj} - \overline{x_j^{(\omega_i)}})(x_{lk} - \overline{x_k^{(\omega_i)}}) \quad j, k = 1, 2, \dots, n$$

式中, l 代表样品在 ω_i 类中的序号,其中 $l = 0, 1, 2, \dots, N_i$;

ω_i 类的协方差矩阵就是

$$\mathbf{S}^i = \begin{pmatrix} s_{11}^i & s_{12}^i & \cdots & s_{1n}^i \\ s_{21}^i & s_{22}^i & \cdots & s_{2n}^i \\ \vdots & \vdots & \vdots & \vdots \\ s_{m1}^i & s_{m2}^i & \cdots & s_{mn}^i \end{pmatrix}$$

- ③ 计算出每一类的协方差矩阵的逆矩阵 \mathbf{S}_i^{-1} 以及协方差矩阵的行列式 $|\mathbf{S}_i|$ 。
- ④ 求出每一类的先验概率

$$P(\omega_i) \approx N_i/N \quad i=0,1,2,\cdots,9$$

⑤ 将各个数值代入判别函数:

$$h_i(\mathbf{X}) = -\frac{1}{2}(\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}})^T \mathbf{S}_i^{-1} (\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}}) + \ln P(\omega_i) - \frac{1}{2} \ln |\mathbf{S}_i|$$

- ⑥ 判别函数最大值所对应类别就是手写数字的类别。

3. 编程代码

```
% 函数名称: bayesleastererror( )
% 参数: sample: 待识别样品特征
% 返回值: y: 待识别样品所属类别
% 函数功能: 最小错误概率的贝叶斯分类器
function y = bayesleastererror( sample )
clc;
load templet pattern;
% 对样品库和待测样品进行主成分分析
[ pcapat, pcasamp ] = pcapro( sample );
temp = 0;
for i = 1:10
    pattern(i).feature = pcapat( :, temp + 1 : temp + pattern(i).num );
    temp = temp + pattern(i).num;
end
s_cov = [ ];
s_inv = [ ];
s_det = [ ];
for i = 1:10
    s_cov(i).dat = cov( pattern(i).feature' ); % 求各类别的协方差矩阵
```

```

s_inv(i).dat = inv(s_cov(i).dat); % 求协方差矩阵的逆矩阵
s_det(i) = det(s_cov(i).dat); % 求协方差矩阵的行列式
end
sum1 = 0;
p = [];
for i = 1:10
    sum1 = sum1 + pattern(i).num; % 求样品库样品总数
end
for i = 1:10
    p(i) = pattern(i).num/sum1; % 求各类别的先验概率
end
h = [];
mean_sap = [];
for i = 1:10
    mean_sap(i).dat = mean(pattern(i).feature)'; % 求每一类样品的特征均值
end
% 计算最大的判别函数
for i = 1:10
    h(i) = (pcasamp-mean_sap(i).dat)' * s_inv(i).dat * (pcasamp-mean_sap(i).dat)...
        * (-0.5) + log(p(i)) + log(abs(s_det(i))) * (-0.5);
end
[maxval maxpos] = max(h);
y = maxpos - 1;

```

4. 效果图

采用基于最小错误率的贝叶斯分类实现效果如图 4-10 所示。



图 4-10 基于最小错误率的贝叶斯分类实现效果

4.7 基于最小风险的贝叶斯分类实现

1. 实现步骤

设待测样品如图 4-11 所示, 基于最小风险的贝叶斯分类步骤如下。

$$\overline{X^{(\omega_i)}} = \frac{1}{N_{iX \in \omega_i}} \sum X = (\overline{x_1^{(\omega_i)}}, \overline{x_2^{(\omega_i)}}, \dots, \overline{x_n^{(\omega_i)}})^T \quad i = 0, 1, 2, \dots, 9$$
$$s_{jk}^i = \frac{1}{N_i - 1} \sum_{l=1}^{N_i} (x_{lj} - \overline{x_j^{(\omega_i)}})(x_{lk} - \overline{x_k^{(\omega_i)}}) \quad j, k = 1, 2, \dots, n$$

6

图 4-11 待测样品

$$\mathbf{S}^i = \begin{pmatrix} s_{11}^i & s_{12}^i & \cdots & s_{1n}^i \\ s_{21}^i & s_{22}^i & \cdots & s_{2n}^i \\ \vdots & \vdots & \vdots & \vdots \\ s_{n1}^i & s_{n2}^i & \cdots & s_{nn}^i \end{pmatrix}$$

④ 求出每一类的先验概率

$$P[i] = -\frac{1}{2}(\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}})^T \mathbf{S}_i^{-1} (\mathbf{X} - \overline{\mathbf{X}^{(\omega_i)}}) + \ln P(\omega_i) - \frac{1}{2} \ln |\mathbf{S}_i|$$

$$\text{loss}[i][j] = \begin{cases} 0, & i=j \\ 1, & i \neq j \end{cases}$$
$$\text{risk}[i] = \sum_{j=0}^9 \text{loss}[i][j]p[j]$$

```
% 函数名称: bayesleastrisk()
% 参数: sample: 待识别样品特征
% 返回值: y: 待识别样品所属类别
% 函数功能: 最小风险的贝叶斯分类器
```



```

function y = bayesleastrisk( sample)
clc;
load templet pattern;
% 对样品库和待测样品进行主成分分析
[ pcapat,pcasamp] = pcapro( sample );
temp = 0;
for i = 1:10
    pattern(i). feature = pcapat( :,temp + 1 :temp + pattern(i). num );
    temp = temp + pattern(i). num;
end
s _ cov = [ ];
s _ inv = [ ];
s _ det = [ ];
for i = 1:10
    s _ cov(i). dat = cov( pattern(i). feature' );% 求各类别的协方差矩阵
    s _ inv(i). dat = inv( s _ cov(i). dat );% 求协方差矩阵的逆矩阵
    s _ det(i) = det( s _ cov(i). dat );% 求协方差矩阵的行列式
end
sum1 = 0;
p = [ ];
for i = 1:10
    sum1 = sum1 + pattern(i). num;% 求样品库样本总数
end
for i = 1:10
    p(i) = pattern(i). num/sum1;% 求各类别的先验概率
end
h = [ ];
mean _ sap = [ ];
for i = 1:10
    mean _ sap(i). dat = mean( pattern(i). feature' );% 求每一类样品的特征均值
end
% 计算各类别的后验概率
for i = 1:10
    h(i) = (pcasamp- mean _ sap(i). dat)' * s _ inv(i). dat * (pcasamp- mean _ sap(i). dat)...
        * ( -0.5) + log( p(i) ) + log( abs( s _ det(i) ) ) * ( -0.5 );
end
loss = ones( 10 ) - diag( diag( ones( 10 ) ) );
risk = 0;
m = [ ];
% 计算最小风险
for i = 1:10
    m = loss( i, : );
    risk(i) = m * h';
end

```

```
end  
[minval minpos] = min(risk);  
y = minpos - 1;
```

3. 效果图

应用基于最小风险的贝叶斯分类器对某次手写数字6进行分类,总风险值如图4-12所示。找出最小损失所对应的类,该类即是待测样品所属的类别。基于最小风险的贝叶斯分类实现效果如图4-13所示。

```
risk =  
  
1.0e+003 *  
  
-3.5175  -3.5371  -3.4734  -3.3968  -3.3172  -3.4478  -3.5626  -0.9495  -3.5039  -3.2165
```

图4-12 基于最小风险的贝叶斯结果显示



图4-13 最小风险的贝叶斯分类实现效果

本章小结

使用贝叶斯决策需要首先对特征空间中的各类样品的分布了解清楚,得到训练集样品总体的分布知识。若能从训练样品估计近似的正态分布,可以按贝叶斯决策方法对分类器进行设计,包括各类先验概率 $P(\omega_i)$ 及类条件概率密度函数,计算出样品的后验概率 $P(\omega_i | \mathbf{X})$,并以此作为产生判别函数的必要数据,设计出相应的判别函数与决策面。一旦待测样品的特征向量值 \mathbf{X} 已知,就可以确定 \mathbf{X} 对各类的后验概率,也就可按相应的准则分类。这种方法称为参数判别方法。

如果这种分布可以用正态分布等描述,那么决策域的判别函数与分界面方程就可用函数的形式确定下来。所以判别函数等的确定取决于样本统计分布的有关知识。参数分类判别方法一般只能用在有统计知识的场合,或能利用训练样本估计出参数的场合。

贝叶斯分类器可以用一般的形式给出数学上严格的分析以证明:在给出某些变量的条件下,能使分类所造成的平均损失最小,或分类决策的风险最小。因此,能计算出分类器的极限性能。贝叶斯决策采用分类器中最重要的指标——错误率作为产生判别函数和决策面的依据,给出了最一般情况下适用的“最优”分类器设计方法,对各种不同的分类器设计技术在理

论上都有指导意义。分类识别中常常会出现错分类的情况,本章讨论了模式识别中经常涉及到的一些问题,例如,在何种情况下会出现错分类,错分类的可能性会有多大等,用概率论的方法分析了造成错分类的原因和分类出错的根源,并说明与哪些因素有关。介绍了贝叶斯决策的基本概念、贝叶斯公式、基于最小错误率的贝叶斯决策、基于最小风险的贝叶斯决策;并介绍了基于二值数据的贝叶斯分类实现方法、基于最小错误率的贝叶斯分类实现方法和基于最小风险的贝叶斯分类实现方法。

习题 4

1. 分类识别中为什么会有错分类? 在何种情况下会出现错分类?
2. 简述贝叶斯决策所讨论的问题。
3. 简述先验概率、类概率密度函数、后验概率三者关系。
4. 简述 $P(\omega_1 | \mathbf{X})$ 和 $P(\omega_2 | \mathbf{X})$ 与 $P(\mathbf{X} | \omega_1)$ 和 $P(\mathbf{X} | \omega_2)$ 的区别。
5. 简述基于最小错误率的贝叶斯分类原则。
6. 写出基于两类问题最小错误率的贝叶斯判别函数形式。
7. 写出基于多类问题最小错误率的贝叶斯判别函数形式。
8. 写出基于多类问题最小风险贝叶斯决策规则判别函数形式。
9. 简述最小风险贝叶斯决策与最小错误率贝叶斯决策之间的关系。
10. 说明基于二值数据的贝叶斯实现方法。
11. 试说明基于最小错误率的贝叶斯实现方法。

第 5 章 判别函数分类器设计

本章要点:

- ☑ 判别函数的基本概念
- ☑ 线性判别函数
- ☑ 线性判别函数的实现
- ☑ 感知器算法
- ☑ 增量校正算法
- ☑ LMSE 验证可分性
- ☑ LMSE 分类算法
- ☑ Fisher 分类
- ☑ 基于核的 Fisher 分类
- ☑ 线性分类器实现分类的局限性
- ☑ 非线性判别函数
- ☑ 势函数法
- ☑ 支持向量机

5.1 判别函数的基本概念

直接使用贝叶斯决策需要首先得到有关样品总体分布的知识,包括各类先验概率 $P(\omega_1)$ 及类条件概率密度函数,计算出样品的后验概率 $P(\omega_1 | \mathbf{X})$,并以此作为产生判别函数的必要数据,设计出相应的判别函数与决策面,这种方法称为判别函数法。它的前提是对特征空间中的各类样品的分布已很清楚,一旦要测试分类样品的特征向量值 \mathbf{X} 已知,就可以确定 \mathbf{X} 对各类的后验概率,也就可按相应的准则计算与分类。所以判别函数等的确定取决于样品统计分布的有关知识。因此,参数分类判别方法一般只能用在有统计知识的场合,或能利用训练样品估计出参数的场合。

由于一个模式通过某种变换映射为一个特征向量后,该特征向量可以理解为特征空间的一个点,在特征空间中,属于一个类的点集,总是在某种程度上与属于另一个类的点集相分离,各个类之间确定可分离的。因此,如果能够找到一个分离函数(线性或非线性函数),把不同类的点集分开,则分类任务就解决了。判别函数法不依赖于条件概率密度的知识,可以理解为通过几何的方法,把特征空间分解为对应于不同类别的子空间。而且呈线性的分离函数,将使计算简化。

假定样品 \mathbf{X} 有两个特征,即 $\mathbf{X} = (x_1, x_2)^T$,每一个样品都对应二维空间中的一个点。每个点属于一类图像,共分三类: $\omega_1, \omega_2, \omega_3$ 。那么待测 \mathbf{X} 属于哪一类呢? 对这个问题就要看它最接近于哪一类,若最接近于 ω_1 则为 ω_1 类,若最接近于 ω_2 则为 ω_2 类,否则就为 ω_3 类。在各类

之间要有一个边界,若能知道各类之间的边界那么就知道待测样品属于哪一类了。所以,要进一步掌握如何去寻找这条分界线。找分界线的方法就是判别函数法,判别函数法结果就是提供一个确定的分界线方程,这个分界线方程叫做判别函数,因此,判别函数描述了各类之间的分界线的具体形式。

判别函数法按照分界函数的形式可以划分为线性判别函数和非线性判别函数两大类。线性分类器由于涉及的数学方法较为简单,在计算机上实现容易,故在模式识别中被广泛应用。但是,这并不意味着,在模式识别中只有线性分类器就足够了。在模式识别的许多问题中,由于线性分类器固有的局限性,它并不能提供理想的识别效果,必须求助于非线性分类器。而且,有些较为简单的非线性分类器,对某些模式识别问题的解决,显得既简单,效果又好。

第2章介绍了提取手写数字样品的特征,建立了样品特征库;本章讨论根据样品特征库建立分类判别函数,用以对待测的手写数字进行分类。在此基础上,本章介绍线性判别函数和非线性判别函数,并介绍它们的实现方法,其中实现线性判别函数分类的方法有感知器算法、增量校正算法、LMSE 分类算法和 Fisher 分类,实现非线性判别函数分类的方法有分段线性函数法、势函数法、基于核的 Fisher 分类、支持向量机。

5.2 线性判别函数

判别函数分为线性判别函数和非线性判别函数。最简单的判别函数是线性判别函数,它是由所有特征量的线性组合构成的。

1. 两类情况

两类分类器框图如图 5-1 所示,根据计算结果的符号将 \mathbf{X} 分类。

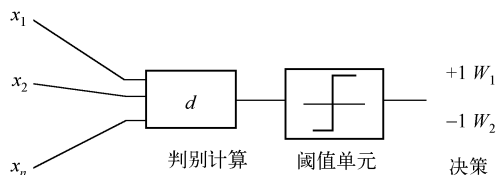


图 5-1 两类情况分类器

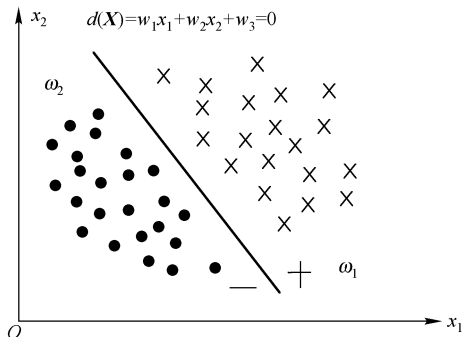


图 5-2 两类模式的线性判别函数

(1) 两个特征

每类模式有两个特征,样品是二维的,在二维模式空间中存在线性判别函数

$$d(\mathbf{X}) = w_1x_1 + w_2x_2 + w_3 = 0 \quad (5-1)$$

式中, w 是参数或者称为权值; x_1, x_2 为坐标变量,即模式的特征值。

可以很明显地看到属于 ω_1 类的任一模式代入 $d(\mathbf{X})$ 后为正值,而属于 ω_2 类的任一模式代入 $d(\mathbf{X})$ 后为负值,如图 5-2 所示。

因此, $d(\mathbf{X})$ 可以用来判断某一模式所属的类别, 在这里我们把 $d(\mathbf{X})$ 称为判别函数。给定某一未知类别的模式 \mathbf{X} , 若 $d(\mathbf{X}) > 0$, 则 \mathbf{X} 属于 ω_1 类; 若 $d(\mathbf{X}) < 0$, 则 \mathbf{X} 属于 ω_2 类; 若 $d(\mathbf{X}) = 0$, 则此时 \mathbf{X} 落在分界线上, 即 \mathbf{X} 的类别处于不确定状态。这一概念不仅局限于两类别情况, 还可推广到有限维欧氏空间中的非线性边界的一般情况。

(2) 三个特征

每类模式有三个特征, 样品是三维的, 判别边界为一平面。

(3) 三个以上特征

每类模式有三个以上特征, 判别边界为一超平面。

对于 n 维空间, 用矢量 $\mathbf{X} = (x_1, x_2, \dots, x_n)^T$ 来表示模式, 一般的线性判别函数形式为

$$d(\mathbf{X}) = w_1 x_1 + w_2 x_2 + \dots + w_n x_n + w_{n+1} = \mathbf{W}_0^T \mathbf{X} + w_{n+1} \quad (5-2)$$

式中, $\mathbf{W}_0 = (w_1, w_2, \dots, w_n)^T$ 称为权矢量或参数矢量。如果在所有模式矢量的最末元素后再附加元素 1, 则式(5-2)可以写成

$$d(\mathbf{X}) = \mathbf{W}^T \mathbf{X} \quad (5-3)$$

的形式。式中 $\mathbf{X} = (x_1, x_2, \dots, x_n, 1)$ 和 $\mathbf{W} = (w_1, w_2, \dots, w_n, w_{n+1})^T$ 分别称为增 1 模式矢量和权矢量。式(5-3)仅仅是为了方便而提出来的, 模式类的基本几何性质并没有改变。

在两种类别情况下, 判别函数 $d(\mathbf{X})$ 有下述性质, 即

$$d(\mathbf{X}) = \mathbf{W}^T \mathbf{X} \begin{cases} > 0, & \mathbf{X} \in \omega_1 \\ < 0, & \mathbf{X} \in \omega_2 \end{cases} \quad (5-4)$$

满足: $d(\mathbf{X}) = \mathbf{W}^T \mathbf{X} = 0$ 的点为两类的判别边界。

2. 多类情况

对于多类别问题, 假设有 M 类模式 $\omega_1, \omega_2, \dots, \omega_M$ 。对于 n 维空间中的 M 个类别, 就要给出 M 个判别函数: $d_1(\mathbf{X}), d_2(\mathbf{X}), \dots, d_M(\mathbf{X})$, 分类器基本形式如图 5-3 所示, 若 \mathbf{X} 属于第 i 类, 则有

$$d_i(\mathbf{X}) > d_j(\mathbf{X}), (j = 1, 2, \dots, M; i \neq j) \quad (5-5)$$

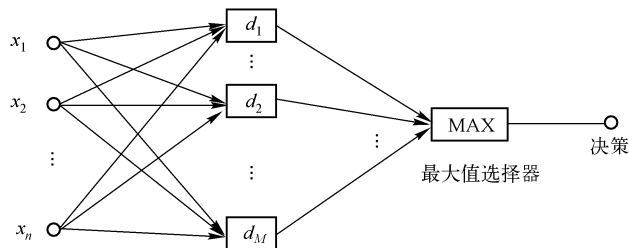


图 5-3 判别函数构成的多类分类器形式

(1) 第一种情况

每一个类别可用单个判别平面分割, 因此 M 类有 M 个判别函数, 具有下面的性质

$$d_i(\mathbf{X}) = \mathbf{W}_i^T(\mathbf{X}) \begin{cases} > 0, & \mathbf{X} \in \omega_i \\ < 0, & \text{其他} \end{cases} \quad i = 1, 2, \dots, M \quad (5-6)$$

如图 5-4 所示,有 3 个模式类,每一类别可用单个判别边界与其余类别划分开。

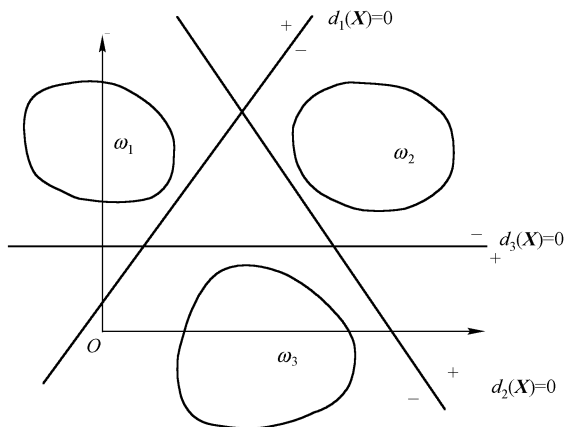


图 5-4 多类情况(1)

例如,手写数字共有 10 个类, $M=10$ 。对任一未知的手写数字,代入判别函数后若只有 $d_7(\mathbf{X}) > 0$,而其他的均小于 0,则该未知的手写数字是 7。

(2) 第二种情况

每两个类别之间可用判别平面分开,有 $M(M-1)/2$ 个判别函数,判别函数形式为

$$d_{ij}(\mathbf{X}) = \mathbf{W}_{ij}^T \mathbf{X}, \text{ 且 } d_{ij}(\mathbf{X}) = -d_{ji}(\mathbf{X}) \quad (5-7)$$

若 $d_{ij}(\mathbf{X}) > 0, \forall j \neq i$, 则 \mathbf{X} 属于 ω_i 类。

没有一个类别可以用一个判别平面与其他类分开,如图 5-5 所示,每一个边界只能分割两类。对于一未知的手写数字,若 $d_{71}, d_{72}, d_{73}, d_{74}, d_{75}, d_{76}, d_{78}, d_{79}, d_{70}$ 均大于 0,则可知该手写数字为 7。

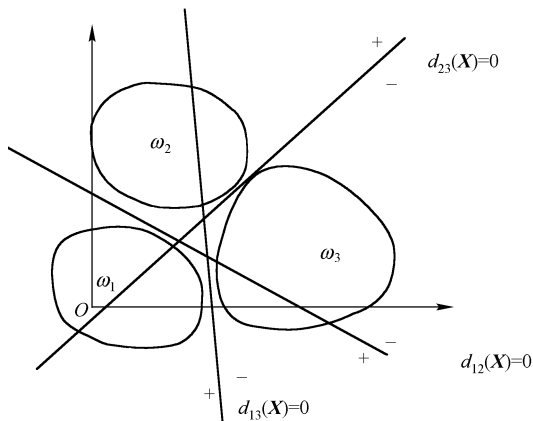


图 5-5 多类情况(2)

(3) 第三种情况

存在 M 个判别函数,判别函数形式为

$$d_i(\mathbf{X}) = \mathbf{W}_i^T \mathbf{X}, i = 1, 2, \cdots, M \tag{5-8}$$

把 \mathbf{X} 代入 M 个判别函数中,判别函数最大的那个类就是 \mathbf{X} 所属类别。与第一种情况的区别在于此种情况下可以有多个判别函数的值大于 0,第一种情况下只有一个判别函数的值大于 0,如图 5-6 所示。

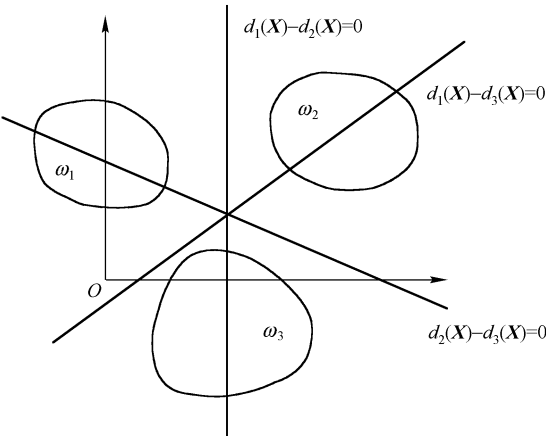


图 5-6 多类情况(3)

对任一未知的手写数字,代入判别函数后若 $d_7(\mathbf{X})$ 为最大值,则该未知的手写数字是 7。

若可用以上几种情况中的任一种线性判别函数来进行分类,则这些模式类称为线性可分的。如表 5-1 所示为线性分类器判别函数形式。

表 5-1 线性分类器判别函数形式

类别情况	判别平面	判别函数形式
<div> 1. 两类情况 <div> <div> <div> <div>x_1</div> <div>x_2</div> <div>x_n</div> </div> <div>判别计算</div> <div> <div> <div>$+1 \ W_1$</div> <div>$-1 \ W_2$</div> </div> <div>决策</div> </div> </div> </div> </div>	<div> 样品是二维的判别边界为一直线 $d(\mathbf{X}) = w_1x_1 + w_2x_2 + w_3 = 0$ </div>	<div> 若 $d(\mathbf{X}) > 0$, 则 $\mathbf{X} \in \omega_1$; 若 $d(\mathbf{X}) < 0$, 则 $\mathbf{X} \in \omega_2$; 若 $d(\mathbf{X}) = 0$, 则 \mathbf{X} 落在分界线上, 类别不确定 </div>
	<div> 样品是三维的判别边界为一平面 $d(\mathbf{X}) = w_1x_1 + w_2x_2 + w_3x_3 + w_4 = 0$ </div>	同上
	<div> 有三个以上特征判别边界为一超平面 $d(\mathbf{X}) = w_1x_1 + w_2x_2 + \cdots + w_nx_n + w_{n+1} = \mathbf{W}_0^T \mathbf{X} + w_{n+1} = 0$ </div>	<div> $d(\mathbf{X}) = \mathbf{W}^T \mathbf{X} \begin{cases} > 0, & \text{当 } \mathbf{X} \in \omega_1 \\ < 0, & \text{当 } \mathbf{X} \in \omega_2 \end{cases}$ $d(\mathbf{X}) = \mathbf{W}^T \mathbf{X} = 0 \text{ 为两类的判别边界}$ </div>

续表

类别情况	判别平面	判别函数形式
<p>2. 多类情况</p>	<p>每一个类别可用单个判别平面分割, M 类有 M 个判别函数 存在不满足条件的不确定区域</p>	$d_i(\mathbf{X}) = \mathbf{W}_i^T \mathbf{X} \begin{cases} > 0, & \mathbf{X} \in \omega_i \\ < 0, & \text{其他} \end{cases}$
	<p>每两个类别之间可用判别平面分开, 有 $M(M-1)/2$ 个判别函数, 存在不满足条件的不确定区域</p>	$d_{ij}(\mathbf{X}) = \mathbf{W}_{ij}^T \mathbf{X}$ <p>若 $d_{ij}(\mathbf{X}) > 0, \forall j \neq i$, 则 \mathbf{X} 属于 ω_i 类</p>
	<p>存在 M 个判别函数 $i = 1, 2, \dots, M$, 除了边界以外, 没有不确定区域, 是第二种情况的特殊状态。在此种条件下可分, 则在第二种情况下也可分; 反之不然</p>	$d_i(\mathbf{X}) = \mathbf{W}_i^T \mathbf{X} \max(d_i(\mathbf{X}))$ <p>把 \mathbf{X} 代入 M 个判别函数中, 判别函数最大的那个类就是 \mathbf{X} 所属类别</p>

模式分类方案取决于两个因素: 判别函数 $d(\mathbf{X})$ 的形式和系数 \mathbf{W} 的确定。前者和所研究模式类的集合形式直接有关。一旦前者确定后, 需要确定的就是后者, 它们可通过模式的样品来确定。

5.3 线性判别函数的实现

前面介绍了判别函数的形式。对于判别函数来说, 应该确定两方面内容: 一方面是方程的形式, 另一方面为方程所带的系数。对于线性判别函数来说, 方程的形式固定为线性, 维数固定为特征向量的维数, 方程组的数量取决于待识别对象的类数。既然方程组的数量、维数和形式已定, 则对判别函数的设计就是确定函数的各系数, 即线性方程的各个权值。下面将讨论怎样确定线性判别函数的系数。

首先按需要确定一准则函数 J , 如 Fisher 准则、感知器算法、增量校正算法、LMSE 算法。确定准则函数 J 达到极值时 \mathbf{W}^* 及 \mathbf{W}_0^* 的具体数值, 从而确定判别函数, 完成分类器设计。线性分类器设计任务是在给定样品集条件下, 确定线性判别函数的各项系数; 对待测样品进行分类时, 能满足相应的准则函数 J 为最优的要求。这种方法的具体过程可大致分为以下几个方面。

① 确定使用的判别函数类型或决策面方程类型, 如线性分类器, 分段线性分类器, 非线性分类器或近邻法等。

② 按需要确定一准则函数 J , 如 Fisher 准则、感知器算法、增量校正算法、LMSE 算法。增量校正算法与感知器算法的实现相似, 只是在进行权矢量修正时加上了权系数; LMSE 算法以最小均方误差作为准则。

③ 确定准则函数 J 达到极值时 \mathbf{W}^* 及 \mathbf{W}_0^* 的具体数值, 从而确定判别函数, 完成分类器设计。

在计算机上确定各权值时采用的是“训练”或“学习”的方法,就是挑选一批已分类的样品,把这批样品输入到计算机的“训练”程序中去,通过多次迭代后,准则函数 J 达到极值,得到正确的线性判别函数。

5.4 感知器算法

1. 理论基础

既然判别函数分类器的训练过程就是确定该函数的权集,为叙述方便起见,我们从判别函数的一般形式着手。对于两类问题来说,设有判别函数

$$d(\mathbf{X}) = w_1 x_1 + w_2 x_2 + w_3 = 0 \quad (5-9)$$

并已知训练集 $\mathbf{X}_A, \mathbf{X}_B, \mathbf{X}_C, \mathbf{X}_D$ 且

$$\{\mathbf{X}_A, \mathbf{X}_B\} \in \omega_1$$

因而对它们来说

$$d(\mathbf{X}) > 0$$

有

$$\{\mathbf{X}_C, \mathbf{X}_D\} \in \omega_2$$

因而对它们来说

$$d(\mathbf{X}) < 0$$

设

$$\mathbf{X}_A = \begin{pmatrix} x_{1A} \\ x_{2A} \end{pmatrix} \quad \mathbf{X}_B = \begin{pmatrix} x_{1B} \\ x_{2B} \end{pmatrix} \quad \mathbf{X}_C = \begin{pmatrix} x_{1C} \\ x_{2C} \end{pmatrix} \quad \mathbf{X}_D = \begin{pmatrix} x_{1D} \\ x_{2D} \end{pmatrix}$$

则判别函数可以联立成

$$\begin{cases} x_{1A}w_1 + x_{2A}w_2 + w_3 > 0 \\ x_{1B}w_1 + x_{2B}w_2 + w_3 > 0 \\ x_{1C}w_1 + x_{2C}w_2 + w_3 < 0 \\ x_{1D}w_1 + x_{2D}w_2 + w_3 < 0 \end{cases}$$

即

$$\begin{cases} x_{1A}w_1 + x_{2A}w_2 + w_3 > 0 \\ x_{1B}w_1 + x_{2B}w_2 + w_3 > 0 \\ -x_{1C}w_1 - x_{2C}w_2 - w_3 > 0 \\ -x_{1D}w_1 - x_{2D}w_2 - w_3 > 0 \end{cases}$$

因此判别函数可写成一般方程形式

$$\mathbf{X}\mathbf{W} > 0 \quad (5-10)$$

式中, \mathbf{W} 为权矢量

$$\mathbf{W} = (w_1, w_2, w_3)^T$$

\mathbf{X} 为这个样品特征值的增 1 矩阵

$$\mathbf{X} = \begin{pmatrix} x_{1A} & x_{2A} & 1 \\ x_{1B} & x_{2B} & 1 \\ -x_{1C} & -x_{2C} & -1 \\ -x_{1D} & -x_{2D} & -1 \end{pmatrix}$$

训练过程就是对判断好的样品集求解权矢量 \mathbf{W} , 即根据已知类别的样品求出权系数, 形成判别界线(面), 再对未知类别的样品求出其类别。这是一个线性联立不等式的求解问题, 只对线性可分问题方程(5-10)才有解。对这样的问题来说, 如果有解, 其解也不一定是单值的, 因而就有一个按不同条件取得最优解的问题。因此出现了多种不同的算法, 这里介绍梯度法。

(1) 梯度下降法

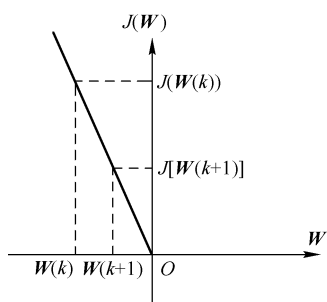


图 5-7 梯度下降法
准则函数示意图

因为求某一函数 $f(\mathbf{W})$ 的数值解, 通常只能求出在某种意义下的最优解, 即先定义一个准则函数, 然后在使此准则函数最大或最小的情况下, 求出 $f(\mathbf{W})$ 的解, 如图 5-7 所示。梯度法就是先确定一准则函数 $J(\mathbf{W})$, 然后选一初值 $\mathbf{W}(1)$, 这样可用迭代式

$$\mathbf{W}(k+1) = \mathbf{W}(k) - C \cdot \Delta J(\mathbf{W}(k)) \quad (5-11)$$

找到 \mathbf{W} 的数值解。

设有一组样品 $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N$, 其中 \mathbf{X}_i 是规范化增广样品向量, 目的是找一个解向量 \mathbf{W}^* , 使得

$$\mathbf{W}^T \mathbf{X}_i > 0 \quad i = 1, 2, \dots, N \quad (5-12)$$

显然, 对于线性可分情况, 该问题才有解。为此, 这里首先考虑处理线性可分问题的算法。可将准则函数 J 的形式选为

$$J(\mathbf{W}, \mathbf{X}) = \alpha (|\mathbf{W}^T \mathbf{X}| - \mathbf{W}^T \mathbf{X}) \quad (5-13)$$

① 当 \mathbf{X} 被错误分类时, 就有 $\mathbf{W}^T \mathbf{X} \leq 0$, 或 $-\mathbf{W}^T \mathbf{X} \geq 0$; 因此, 式(5-13)中的 $J(\mathbf{W}, \mathbf{X})$ 总是大于 0。因此, 在 \mathbf{X} 被错误分类时, 函数 $J(\mathbf{W}, \mathbf{X}) = J(\mathbf{W})$ 的坐标原点左侧, 为一反比于 \mathbf{W} 的斜线。

② 在 $\mathbf{W}^T \mathbf{X} > 0$ 时, 有 $J(\mathbf{W}, \mathbf{X}) = J_{\min}(\mathbf{W}, \mathbf{X}) = 0$, \mathbf{X} 被正确分类时, 在坐标原点右侧 $J(\mathbf{W}, \mathbf{X}) = 0$, \mathbf{W} 获得一确定解的区域。

矢量的方向主要是由其取值最大的分量决定的, 故负梯度向量 $-\Delta J(\mathbf{W})$ 指出了 \mathbf{W} 的最陡下降方向。当梯度向量为 0 时, 达到了函数的极值。对联立不等式(5-10)求解 \mathbf{W} 的问题, 可转化为求函数极小值的问题。若准则函数 J 的梯度 $\frac{\partial J}{\partial \mathbf{W}}$ 收敛为 0, 则表明达到 J 的极值, 从而就是方程(5-10)的最优解 \mathbf{W} 。

我们把样品集看做一个不断重复出现的序列而逐个加以考虑。例如, 有三个样品组成的样品序列 $\bar{y}_1, \bar{y}_2, \bar{y}_3, \bar{y}_1, \bar{y}_2, \bar{y}_3, \bar{y}_1, \bar{y}_2, \dots$, 其中画“ $\bar{}$ ”的是被分错的样品, 则把错分样品序列 $y_1, y_3, y_1, y_2, y_1, \dots$ 记做 $y^1, y^2, y^3, y^4, y^5, \dots$; 对于任意权向量 $\mathbf{W}(k)$, 如果它把某个样品分错了, 则对 $\mathbf{W}(k)$ 做一次修正。这种方法称为单样品修正法。由于仅在发现分错类时, 才修正 $\mathbf{W}(k)$, 所以只注意那些被分错类的样品就行了。当且仅当错分样品集合为空集时, $J(\mathbf{W}^*) = \min J$

$(\mathbf{W}) = 0$, 这时将不存在错分样品, 这时的 \mathbf{W} 就是我们要寻找的解向量 \mathbf{W}^* 。

有了准则函数 $J(\mathbf{W})$, 下一步便是求使 $J(\mathbf{W})$ 达到极小值时的解向量 \mathbf{W}^* 了。设 $\mathbf{W}(k)$ 为 \mathbf{W} 的 k 次迭代解, $\mathbf{W}(k+1)$ 为其后的另一迭代解, 只要下一迭代解沿斜线下降, 总有可能收敛地搜索到满足要求的 \mathbf{W} 值, 即有

$$\mathbf{W}(k+1) = \mathbf{W}(k) - C \left\{ \frac{\partial J}{\partial \mathbf{W}(k)} \right\}_{\mathbf{W}=\mathbf{W}(k)} \quad (5-14)$$

式中, C 为有助于收敛的校正系数。

在确定式(5-13)的系数 α 后, 可推导出迭代算法的具体关系。不妨设 $\alpha = 1/2$, 即

$$J(\mathbf{W}, \mathbf{X}) = \frac{1}{2} (|\mathbf{W}^T \mathbf{X}| - \mathbf{W}^T \mathbf{X}) \quad (5-15)$$

则

$$\frac{\partial J}{\partial \mathbf{W}} = \frac{1}{2} [\mathbf{X} \operatorname{sgn}(\mathbf{W}^T \mathbf{X}) - \mathbf{X}] \quad (5-16)$$

其中

$$\operatorname{sgn}(\mathbf{W}^T \mathbf{X}) = \begin{cases} 1, & \text{若 } \mathbf{W}^T \mathbf{X} > 0 \\ -1, & \text{其他} \end{cases} \quad (5-17)$$

将式(5-16)代入式(5-14)后, 有

$$\begin{aligned} \mathbf{W}(k+1) &= \mathbf{W}(k) + \frac{C}{2} \{ \mathbf{X}(k) - \mathbf{X}(k) \operatorname{sgn}[\mathbf{W}^T(k) \mathbf{X}(k)] \} \\ &= \begin{cases} \mathbf{W}(k), & \text{若 } \mathbf{W}^T(k) \mathbf{X}(k) > 0 \\ \mathbf{W}(k) + C\mathbf{X}(k), & \text{其他} \end{cases} \end{aligned} \quad (5-18)$$

即 $\mathbf{W}^T(k) \mathbf{X}(k) > 0$ 时, 表明对样品正确分类, $\mathbf{W}(k+1) = \mathbf{W}(k)$, 不做修正; 反之, 当 $\mathbf{W}^T(k) \mathbf{X}(k) \leq 0$ 时, 表明对样品错误分类, $\mathbf{W}(k+1) = \mathbf{W}(k) + C\mathbf{X}(k)$, 即应添加一修正项 $C\mathbf{X}(k)$ 。这就是感知器算法的基本形式。

梯度下降法可以简单地叙述为: 任意给定初始权向量 $\mathbf{W}(1)$, 第 $k+1$ 次迭代时的权向量 $\mathbf{W}(k+1)$ 等于第 k 次的权向量加上被 $\mathbf{W}(k)$ 错分的样品乘以某个系数 C , 可以证明, 对于线性可分的样品集, 经过有限次修正, 一定可以找到一个解向量 \mathbf{W}^* , 即算法在有限步内收敛。不失一般性, 可令 $C=1$ 。其收敛速度的快慢取决于初始权向量 $\mathbf{W}(1)$ 和系数 C 。

(2) “奖惩”算法

将感知器算法进一步具体化就是“奖惩”算法。

➤ 两类情况

若 $\mathbf{X}(k) \in \omega_1$ 且 $\mathbf{W}^T(k) \mathbf{X}(k) > 0$, 或 $\mathbf{X}(k) \in \omega_2$ 且 $\mathbf{W}^T(k) \mathbf{X}(k) < 0$, 则不须修正, 即 $\mathbf{W}(k+1) = \mathbf{W}(k)$ 。反之, 则须修正, 即

若 $\mathbf{X}(k) \in \omega_1$ 且 $\mathbf{W}^T(k) \mathbf{X}(k) \leq 0$, 则 $\mathbf{W}(k+1) = \mathbf{W}(k) + C\mathbf{X}(k)$

若 $\mathbf{X}(k) \in \omega_2$ 且 $\mathbf{W}^T(k) \mathbf{X}(k) \geq 0$, 则 $\mathbf{W}(k+1) = \mathbf{W}(k) - C\mathbf{X}(k)$

➤ 多类情况

手写体数字的识别属于多类情况。对于多类问题, 我们可以采用判别函数最大值的方法来训练, 即对于 M 个类, 应该有 M 个判别函数。

设有 $\omega_1, \omega_2, \dots, \omega_M$ (共 M 类样品), 并且在 k 次迭代时出现的样品 $\mathbf{X}(k) \in \omega_i$, 如果采用奖

惩算法,则 M 个判别函数 $d_i[\mathbf{X}(k) = \mathbf{W}_i^T(k)\mathbf{X}(k), i = 1, 2, \dots, M]$ 都应加以计算。

① 对于 $d_i[\mathbf{X}(k)] > d_j[\mathbf{X}(k)]$, 则权矢量不必加以修正, 即

$$\begin{cases} \mathbf{W}_i(k+1) = \mathbf{W}_i(k) \\ \mathbf{W}_j(k+1) = \mathbf{W}_j(k) \end{cases} \quad (5-19)$$

② 若对于 $d_i[\mathbf{X}(k)] \leq d_j[\mathbf{X}(k)]$, 则应按下式修正权矢量:

$$\begin{cases} \mathbf{W}_i(k+1) = \mathbf{W}_i(k) + C\mathbf{X}(k) \\ \mathbf{W}_j(k+1) = \mathbf{W}_j(k) - C\mathbf{X}(k) \end{cases} \quad (5-20)$$

例如, 手写数字分 10 个类, 每个样品有 25 个特征, 有 10 个判别函数如下:

$$\begin{aligned} d_0(\mathbf{X}) &= \omega_{00}x_0 + \omega_{01}x_1 + \dots + \omega_{0,24}x_{24} \\ d_1(\mathbf{X}) &= \omega_{10}x_0 + \omega_{11}x_1 + \dots + \omega_{1,24}x_{24} \\ &\vdots \\ d_9(\mathbf{X}) &= \omega_{90}x_0 + \omega_{91}x_1 + \dots + \omega_{9,24}x_{24} \end{aligned}$$

对于某一个已知类别为 1 的样品 \mathbf{X} , 计算这 10 个判别函数, 如果有 $d_1(\mathbf{X}) > d_j(\mathbf{X}), (j = 0, 2, 3, \dots, 9)$, $\mathbf{W}_j(j=0, 2, 3, \dots, 9)$ 不修改, $\mathbf{W}_j(k+1) = \mathbf{W}_j(k)$ 。

而若 $d_1(\mathbf{X}) < d_8(\mathbf{X})$ 或 $d_1(\mathbf{X}) < d_9(\mathbf{X})$, 则 $\mathbf{W}_1, \mathbf{W}_2$ 至 \mathbf{W}_9 按如下规则修改

$$\begin{cases} \mathbf{W}_1(k+1) = \mathbf{W}_1(k) + C\mathbf{X}(k) \\ \mathbf{W}_j(k+1) = \mathbf{W}_j(k) - C\mathbf{X}(k) \end{cases} \quad (j=0, 2, 3, \dots, 9)$$

(3) 实例说明

为了理解算法的思路, 这里给一个简单的例子, 用感知器算法对三类模式求判别函数, 每类仅含一个样品:

$$\omega_1: \{(0, 0)^T\}, \omega_2: \{(1, 1)^T\}, \omega_3: \{(-1, 1)^T\}。$$

其增广形式为

$$(0, 0, 1)^T, (1, 1, 1)^T, (-1, 1, 1)^T。$$

取参数 $C=1$, 初始值 $\mathbf{W}_1(1) = \mathbf{W}_2(1) = \mathbf{W}_3(1) = (0, 0, 0)^T$ 。其步骤如下。

① 输入样品 $\mathbf{X}(1)$, 则

$$\begin{aligned} d_1[\mathbf{X}(1)] &= \mathbf{W}_1^T(1)\mathbf{X}(1) = 0 \\ d_2[\mathbf{X}(1)] &= \mathbf{W}_2^T(1)\mathbf{X}(1) = 0 \\ d_3[\mathbf{X}(1)] &= \mathbf{W}_3^T(1)\mathbf{X}(1) = 0 \end{aligned}$$

因 $\mathbf{X}(1) \in \omega_1$, 而 $d[\mathbf{X}(1)]$ 均为 0, $d_1[\mathbf{X}(1)]$ 不是最大值, 故需根据式 (5-20) 来修改权矢量

$$\begin{aligned} \mathbf{W}_1(2) &= \mathbf{W}_1(1) + \mathbf{X}(1) = (0, 0, 1)^T \\ \mathbf{W}_2(2) &= \mathbf{W}_2(1) - \mathbf{X}(1) = (0, 0, -1)^T \\ \mathbf{W}_3(2) &= \mathbf{W}_3(1) - \mathbf{X}(1) = (0, 0, -1)^T \end{aligned}$$

② 输入 $\mathbf{X}(2), \mathbf{X}(2) \in \omega_2$, 则

$$\mathbf{W}_1^T(2)\mathbf{X}(2) = 1, \mathbf{W}_2^T(2)\mathbf{X}(2) = -1, \mathbf{W}_3^T(2)\mathbf{X}(2) = -1$$

$d_2[\mathbf{X}(2)]$ 不是最大值, 故需修改权矢量

$$\mathbf{W}_1(3) = \mathbf{W}_1(2) - \mathbf{X}(2) = (-1, -1, 0)^T$$

$$\mathbf{W}_2(3) = \mathbf{W}_2(2) + \mathbf{X}(2) = (1, 1, 0)^T$$

$$\mathbf{W}_3(3) = \mathbf{W}_3(2) - \mathbf{X}(2) = (-1, -1, -2)^T$$

③ 输入样品 $\mathbf{X}(3)$, $\mathbf{X}(3) \in \omega_3$, 则

$$\mathbf{W}_1^T(3)\mathbf{X}(3) = 0, \mathbf{W}_2^T(3)\mathbf{X}(3) = 0, \mathbf{W}_3^T(3)\mathbf{X}(3) = -2$$

$d_3[\mathbf{X}(3)]$ 不是最大值, 故需修正权矢量

$$\mathbf{W}_1(4) = \mathbf{W}_1(3) - \mathbf{X}(3) = (0, -2, -1)^T$$

$$\mathbf{W}_2(4) = \mathbf{W}_2(3) - \mathbf{X}(3) = (2, 0, -1)^T$$

$$\mathbf{W}_3(4) = \mathbf{W}_3(3) + \mathbf{X}(3) = (-2, 0, -1)^T$$

至此已完成一次迭代, 但未得到对三类模式均为正确的权矢量, 故令 $\mathbf{X}(4) = \mathbf{X}(1)$, $\mathbf{X}(5) = \mathbf{X}(2)$, $\mathbf{X}(6) = \mathbf{X}(3)$, 再次进行迭代。

④ 输入样品 $\mathbf{X}(4)$, 得到

$$\mathbf{W}_1^T(4)\mathbf{X}(4) = -1, \mathbf{W}_2^T(4)\mathbf{X}(4) = -1, \mathbf{W}_3^T(4)\mathbf{X}(4) = -1$$

因 $\mathbf{X}(4) \in \omega_1$, 而 $d_1[\mathbf{X}(4)]$ 不是最大值, 应修正权矢量

$$\mathbf{W}_1(5) = \mathbf{W}_1(4) + \mathbf{X}(4) = (0, -2, 0)^T$$

$$\mathbf{W}_2(5) = \mathbf{W}_2(4) - \mathbf{X}(4) = (2, 0, -2)^T$$

$$\mathbf{W}_3(5) = \mathbf{W}_3(4) - \mathbf{X}(4) = (-2, 0, -2)^T$$

⑤ 输入样品 $\mathbf{X}(5) \in \omega_2$, 各判别函数的值为

$$\mathbf{W}_1^T(5)\mathbf{X}(5) = -2, \mathbf{W}_2^T(5)\mathbf{X}(5) = 0, \mathbf{W}_3^T(5)\mathbf{X}(5) = -4$$

$d_2[\mathbf{X}(5)]$ 是最大值, 这一判别结果是正确的, 因而权矢量不需修正:

$$\mathbf{W}_1(6) = \mathbf{W}_1(5) = (0, -2, 0)^T$$

$$\mathbf{W}_2(6) = \mathbf{W}_2(5) = (2, 0, -2)^T$$

$$\mathbf{W}_3(6) = \mathbf{W}_3(5) = (-2, 0, -2)^T$$

⑥ 输入样品 $\mathbf{X}(6) \in \omega_3$, 判别结果为

$$\mathbf{W}_1^T(6)\mathbf{X}(6) = -2, \mathbf{W}_2^T(6)\mathbf{X}(6) = -4, \mathbf{W}_3^T(6)\mathbf{X}(6) = 0$$

判别结果正确, 因而

$$\mathbf{W}_1(7) = \mathbf{W}_1(6) = (0, -2, 0)^T$$

$$\mathbf{W}_2(7) = \mathbf{W}_2(6) = (2, 0, -2)^T$$

$$\mathbf{W}_3(7) = \mathbf{W}_3(6) = (-2, 0, -2)^T$$

⑦ 输入样品 $\mathbf{X}(7) = (0, 0, 1)^T$, $\mathbf{X}(7) \in \omega_1$, 各判别函数的值为

$$\mathbf{W}_1^T(7)\mathbf{X}(7) = 0, \mathbf{W}_2^T(7)\mathbf{X}(7) = -2, \mathbf{W}_3^T(7)\mathbf{X}(7) = -2$$

至此, 所有的样品都通过了检验, 因而权矢量为

$$\mathbf{W}_1(8) = \mathbf{W}_1(7) = (0, -2, 0)^T$$

$$\mathbf{W}_2(8) = \mathbf{W}_2(7) = (2, 0, -2)^T$$

$$\mathbf{W}_3(8) = \mathbf{W}_3(7) = (-2, 0, -2)^T$$

于是三个判别函数为

$$d_1(\mathbf{X}) = -2x_2$$

$$d_2(\mathbf{X}) = 2x_1 - 2$$

$$d_3(\mathbf{X}) = -2x_1 - 2$$

2. 实现步骤

① 设各个权矢量的初始值为 0, 即 $\mathbf{W}_0 = \mathbf{W}_1 = \mathbf{W}_2 = \cdots = \mathbf{W}_9 = 0$ 。

② 第 k 次输入一个样品 $\mathbf{X}(k)$, 计算第 k 次迭代计算的结果为

$$d_i[\mathbf{X}(k)] = \mathbf{W}_i^T(k) \mathbf{X}(k), i=0, 1, \cdots, 9$$

③ 若 $\mathbf{X}(k) \in \omega_i, i=0, 1, \cdots, 9$, 判断 $d_i[\mathbf{X}(k)]$ 是不是最大值。若是, 则各个权值不需修正; 否则, 各权值需修正:

$$\mathbf{W}_i(k+1) = \mathbf{W}_i(k) + \mathbf{X}(k), \mathbf{W}_j(k+1) = \mathbf{W}_j(k) - \mathbf{X}(k) \quad j=0, 1, \cdots, 9, j \neq i$$

④ 循环执行第②步, 直到输入所有的样品, 权值都不需要修正为止。注意, 对于判别函数分类器样品总数不需要过多。

3. 编程代码

```
%%%%%%%%%%%%%%
% 函数名称:jiangcheng()
% 参数:sample:待识别样品特征
% 返回值: y:待识别样品所属类别
% 函数功能:奖惩算法
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = jiangcheng( sample)

    clc;
    load templet pattern;
    w = zeros(26,10); % 初始化权矢量矩阵
    d = [];
    maxpos = 0;
    maxval = 0;
    f = 1;
    n = []; m = [];
    % 依次输入样本
    for j = 1:100
        for i = 1:10
            f = 1;
            pattern(i).feature(26,j) = 1; % 最后一位置 1
            for k = 1:10
                m = pattern(i).feature(:,j);
                d(k) = w(:,k)' * m;
            end
            % 判断是否为最大值,如果是,f=1,否则 f=0;
            for k = 1:10
                if k ~= i
                    if d(i) <= d(k)
                        f = 0;
                    end
                end
            end
        end
    end
```

```

        end
    end
end
%修正权值
if ~f
    for k = 1:10
        if k == i
            w(:,k) = w(:,k) + pattern(i).feature(:,j);
        else
            w(:,k) = w(:,k) - pattern(i).feature(:,j);
        end
    end
end
end
end
sample(26) = 1;
h = [];
%计算各类别的判别函数
for k = 1:10
    h(k) = w(:,k)' * sample';
end
[maxval,maxpos] = max(h);
y = maxpos - 1;

```

4. 效果图

奖惩算法识别效果如图 5-8 所示。



图 5-8 奖惩算法识别效果

由于模式识别算法复杂,步骤较多,实现起来有一定的难度。为了使样品库少一些,将精力着重放在算法的理解及编程实现上,我们将模板设计为 5×5 , 比较小。读者可以将模板扩充更大一些,这样计算会更准确些。

5.5 增量校正算法

1. 理论基础

(1) 回归函数

从随机估计的角度来看,对于 M 类问题来说,贝叶斯分类器的判别函数可以表达成

$$d_i(\mathbf{X}) = P(\omega_i | \mathbf{X})P(\mathbf{X}) \quad i = 1, 2, \dots, M$$

认为 $P(\mathbf{X})$ 与类别无关,因而

$$d_i(\mathbf{X}) = P(\omega_i | \mathbf{X}) \quad (5-21)$$

如果 $\mathbf{X} \in \omega_i$, 则

$$d_i(\mathbf{X}) > d_j(\mathbf{X}) \quad \forall j = 1, 2, \dots, M \text{ 且 } j \neq i$$

对于两类问题来说,其判别边界为

$$d_1(\mathbf{X}) - d_2(\mathbf{X}) = 0 \quad (5-22)$$

即

$$d(\mathbf{X}) = d_1(\mathbf{X}) - d_2(\mathbf{X}) = P(\omega_1 | \mathbf{X}) - [1 - P(\omega_1 | \mathbf{X})] = 2P(\omega_1 | \mathbf{X}) - 1 = 0$$

则有

$$\begin{aligned} d(\mathbf{X}) > 0 &\Rightarrow \begin{cases} \mathbf{X} \in \omega_1 \\ \mathbf{X} \in \omega_2 \end{cases} \\ d(\mathbf{X}) \leq 0 &\Rightarrow \begin{cases} \mathbf{X} \in \omega_1 \\ \mathbf{X} \in \omega_2 \end{cases} \end{aligned} \quad (5-23)$$

或者

$$\begin{aligned} P(\omega_1 | \mathbf{X}) > 1/2 &\Rightarrow \begin{cases} \mathbf{X} \in \omega_1 \\ \mathbf{X} \in \omega_2 \end{cases} \\ P(\omega_1 | \mathbf{X}) \leq 1/2 &\Rightarrow \begin{cases} \mathbf{X} \in \omega_1 \\ \mathbf{X} \in \omega_2 \end{cases} \end{aligned} \quad (5-24)$$

$1/2$ 只是对判别界限的一个位移,因而为分析方便起见,可把判别函数直接看做

$$d(\mathbf{X}) = P(\omega_1 | \mathbf{X}) = 0 \quad (5-25)$$

当然,如果为两类问题,对于最后求得的判别函数应该考虑进行判别界限的这一位移。

由于多项式处理问题的方便性,在随机估计中仍以一多项式来逼近函数 $P(\omega_i | \mathbf{X})$, 即令

$$P(\omega_i | \mathbf{X}) = \sum_{j=1}^{n+1} w_{ij} \varphi_j(\mathbf{X})$$

式中, $\varphi_j(\mathbf{X})$ 为一定形式的非线性或线性方程。由于 $\varphi_j(\mathbf{X})$ 中的各项最终总是可以以一个线性变量来代替,因而可写出

$$P(\omega_i | \mathbf{X}) = \mathbf{W}_i^T \mathbf{X} \quad (5-26)$$

式中, $\mathbf{W}_i = [w_{i1}, w_{i2}, \dots, w_{i(n+1)}]^T$ 为权矢量, $\mathbf{X} = (x_1, x_2, \dots, x_n, 1)^T$ 为特征增 1 矢量。现在的问题是通过对样品的逐个观察来估计权矢量的。

当各类样品以 $P(\omega_i | \mathbf{X})$ 的概率密度出现时,实际观测值不是概率密度而是逻辑判断结论

$$r_i(\mathbf{X}) = \begin{cases} 1, & \mathbf{X} \in \omega_i \\ 0, & \mathbf{X} \notin \omega_i \end{cases}$$

可以把这种观察结果看做由于系统中引入了均值为零的随机干扰 η 所造成的后果,如图 5-9 所示。

这种干扰的影响可写成

$$r_i(\mathbf{X}) = P(\omega_i | \mathbf{X}) + \eta \quad (5-27)$$

因为

$$E(\eta) = 0$$

所以可写出 $P(\omega_i | \mathbf{X})$ 的估计为

$$\hat{P}(\omega_i | \mathbf{X}) = E\{P(\omega_i | \mathbf{X})\} = E\{r_i(\mathbf{X})\}$$

按照统计学上的概念, $\hat{P}(\omega_i | \mathbf{X})$ 称为 $P(\omega_i | \mathbf{X})$ 的回归估计, $\hat{P}(\omega_i | \mathbf{X})$ 也称为回归函数。

如图 5-10 所示, 这种回归函数 $\hat{h}(\mathbf{X})$ 与观察结果 $h(\mathbf{X})$ 之间的关系, 由于受到随机干扰的影响, 观测值散布在回归函数 $\hat{h}(\mathbf{X})$ 的四周并形成误差。从理论上来说, 如果在变量 \mathbf{X} 的每一点上测量大量的观测值, 则每一点上 $h(\mathbf{X})$ 的期望值应该就是回归估计。但因为无法按照这样的要求来选取样品, 只能按照有限的观察次数来进行估计。

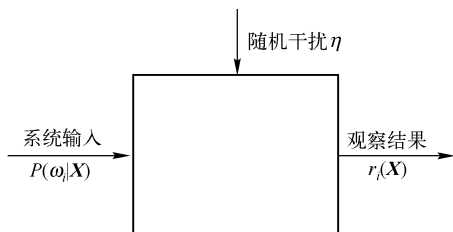


图 5-9 对 $P(\omega_i | \mathbf{X})$ 的观察结果

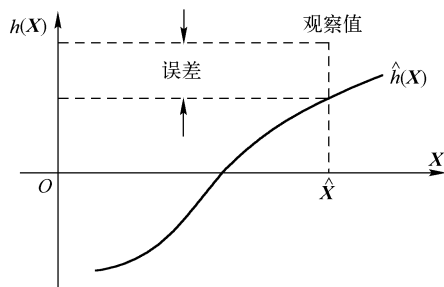


图 5-10 回归函数 $\hat{h}(\mathbf{X})$ 与观察结果 $h(\mathbf{X})$

如果再考虑到式(5-26)的关系, 则可看到所希望得到的权矢量的估计值 $\hat{\mathbf{W}}_i$ 。这就是说, 从式(5-26)来看, \mathbf{W}_i 也是以随机变量的角色出现的。所研究的是对 $P(\omega_i | \mathbf{X})$ 的估计, 实际上希望解决的是对 \mathbf{W}_i 的估计。

一个算法总是以某一个准则函数出现的最佳状态作为条件而建立起来的, 设准则函数为

$$J(\mathbf{W}_i, \mathbf{X}) = E\{f(\mathbf{W}_i, \mathbf{X})\} \quad (5-28)$$

则出现最优的条件为

$$\partial J / \partial \mathbf{W} = f'(\mathbf{W}_i) = 0$$

由此确定了相应于最优状态时的 \mathbf{W}_i 值。如果 $\partial J / \partial \mathbf{W}_i$ 是一个回归函数, 则由 $f'(\mathbf{W}_i) = 0$ 解 \mathbf{W}_i , 也就是对回归函数 $\partial J / \partial \mathbf{W} = 0$ 求其根 \mathbf{W}_i 。从图 5-10 来说, 就是对 $\hat{h}(\mathbf{X}) = 0$ 求根 $\hat{\mathbf{X}}$ 。这样一个随机估计问题就是对回归函数解它等于零时的根。由于 \mathbf{W}_i 为随机变量以及准则函数所采用的数学期望的形式, 使得 $\partial J / \partial \mathbf{W}$ 成为回归函数的假设可以成立。

为便于讨论起见, 把回归函数 $\partial J / \partial \mathbf{W}$ 写成 $g(\mathbf{W})$, 把它的观测值写成 $h(\mathbf{W})$, 认为函数 $g(\mathbf{W})$ 只具有单根 $\hat{\mathbf{W}}$, 因而 $g(\hat{\mathbf{W}}) = 0$ 。按照对噪声统计性质的规定有 $E\{h(\mathbf{W})\} = g(\mathbf{W})$ 。另外, 还认为观测值对其数学期望的方差处在一定的有限范围之内, 即

$$\sigma^2 = E\{[g(\mathbf{W}) - h(\mathbf{W})]^2\} \quad (5-29)$$

对所有 \mathbf{W} 值来说, 都是一个有限的值。从而我们引入了 Robbins-Monro (R-M) 算法, R-M 算法建立在下述递推方程的基础上:

$$\mathbf{W}(k+1) = \mathbf{W}(k) - \alpha_k h[\mathbf{W}(k)] \quad (5-30)$$

$\mathbf{W}(k+1)$ 和 $\mathbf{W}(k)$ 是 \mathbf{W} 的 $k+1$ 次和 k 次递推值, α_k 是一正级数中的第 k 个分量。 α_k 按照调和级数 $1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \dots, \frac{1}{k}$ (k 为递推次数) 顺次递减。

现在要把 R-M 算法应用到模式识别上来, 因为 $g(\mathbf{W}) = E\{h(\mathbf{W})\}$, 也即

$$\partial J / \partial \mathbf{W} = E\{h(\mathbf{W})\} \quad (5-31)$$

J 为准则函数

$$J(\mathbf{W}, \mathbf{X}) = E\{f(\mathbf{W}, \mathbf{X})\} \quad (5-32)$$

对式(5-32)进行偏微分, 可得

$$\frac{\partial J(\mathbf{W}, \mathbf{X})}{\partial \mathbf{W}} = E\left\{\frac{\partial f(\mathbf{W}, \mathbf{X})}{\partial \mathbf{W}}\right\} \quad (5-33)$$

对比一下式(5-31)和式(5-33)可得

$$h(\mathbf{W}) = \frac{\partial f(\mathbf{W}, \mathbf{X})}{\partial \mathbf{W}} \quad (5-34)$$

因而在模式识别情况下, R-M 算法可以表示成

$$\mathbf{W}(k+1) = \mathbf{W}(k) - \alpha_k \left\{ \frac{\partial f(\mathbf{W}, \mathbf{X})}{\partial \mathbf{W}} \right\}_{\mathbf{W}=\mathbf{W}(k)} \quad (5-35)$$

在 $\frac{\partial f(\mathbf{W}, \mathbf{X})}{\partial \mathbf{W}} = 0$ 时得到回归函数的根 $\hat{\mathbf{W}}$, 而这时

$$\mathbf{W}(k+1) = \mathbf{W}(k)$$

现在对式(5-35)的求解就决定于准则函数的具体形式了。

(2) 增量校正算法

增量校正算法采用绝对偏差的平均值为最小的原则, 这时准则函数取为

$$J(\mathbf{W}_i, \mathbf{X}) = E\{|r_i(\mathbf{X}) - \mathbf{W}_i^T \mathbf{X}|\} \quad (5-36)$$

式中

$$r_i(\mathbf{X}) = \begin{cases} 1, & \mathbf{X} \in \omega_i \\ 0, & \mathbf{X} \notin \omega_i \end{cases}$$

式(5-36)对 \mathbf{W}_i 进行偏微分, 可得

$$\frac{\partial J}{\partial \mathbf{W}_i} = E\{-\mathbf{X} \operatorname{sgn}[r_i(\mathbf{X}) - \mathbf{W}_i^T \mathbf{X}]\} \quad (5-37)$$

式中

$$\operatorname{sgn}[r_i(\mathbf{X}) - \mathbf{W}_i^T \mathbf{X}] = \begin{cases} 1, & r_i(\mathbf{X}) \geq \mathbf{W}_i^T \mathbf{X} \\ -1, & r_i(\mathbf{X}) < \mathbf{W}_i^T \mathbf{X} \end{cases}$$

由式(5-33)可以写出

$$\frac{\partial f(\mathbf{W}_i, \mathbf{X})}{\partial \mathbf{W}_i} = -\mathbf{X} \operatorname{sgn}[r_i(\mathbf{X}) - \mathbf{W}_i^T \mathbf{X}] \quad (5-38)$$

因而算法方程为

$$\mathbf{W}_i(k+1) = \mathbf{W}_i(k) + \alpha_k \mathbf{X}(k) \operatorname{sgn}\{r_i[\mathbf{X}(k)] - \mathbf{W}_i^T(k) \mathbf{X}(k)\} \quad (5-39)$$

\mathbf{W}_i 的初值 $\mathbf{W}(1)$ 是可以任意选取的, 式(5-39)也可写成

$$\mathbf{W}_i(k+1) = \begin{cases} \mathbf{W}_i(k) + \alpha_k \mathbf{X}(k), & \mathbf{W}_i^T(k) \mathbf{X}(k) < r_i[\mathbf{X}(k)] \\ \mathbf{W}_i(k) - \alpha_k \mathbf{X}(k), & \mathbf{W}_i^T(k) \mathbf{X}(k) \geq r_i[\mathbf{X}(k)] \end{cases} \quad (5-40)$$

算法的每一步都要校正权向量。而感知器算法只有在某一个模式样品被错误分类时才校正。由式(5-39)或式(5-40)看出,算法每一步的校正值都正比于增量 α_k ,故称为增量校正算法。

对于多类问题来说,判别函数为

$$d_i(\mathbf{X}) = \mathbf{W}_i^T \mathbf{X}$$

如果 $\mathbf{X} \in \omega_i$, 则

$$d_i(\mathbf{X}) = \mathbf{W}_i^T \mathbf{X} = r_i[\mathbf{X}] = 1$$

而

$$d_j(\mathbf{X}) = \mathbf{W}_j^T \mathbf{X} = r_j[\mathbf{X}] = 0$$

(3) 实例说明

为了说明该算法的实现过程,这里给出一个例子,给定两类样品:

$$\omega_1: (0,0,0)^T, (1,0,0)^T, (1,0,1)^T, (1,1,0)^T$$

$$\omega_2: (0,0,1)^T, (0,1,0)^T, (0,1,1)^T, (1,1,1)^T$$

现在用增量校正算法计算判别函数。在进行迭代运算之前,各样品的特征向量经过增1:

$$\omega_1: (0,0,0,1)^T, (1,0,0,1)^T, (1,0,1,1)^T, (1,1,0,1)^T$$

$$\omega_2: (0,0,1,1)^T, (0,1,0,1)^T, (0,1,1,1)^T, (1,1,1,1)^T$$

以 $\mathbf{W}(1) = \mathbf{0}, \alpha_k = 1/k$ 进行讨论。

① 令 $\mathbf{X}(1) = (0,0,0,1)^T, \mathbf{X}(1) \in \omega_1, \mathbf{W}(2) = \mathbf{W}(1) + \alpha_1 \mathbf{X}(1) \operatorname{sgn}\{r[\mathbf{X}(1)] - \mathbf{W}^T(1)\mathbf{X}(1)\}$

因为 $\alpha_1 = 1, r[\mathbf{X}(1)] = 1$ 和 $r[\mathbf{X}(1)] > \mathbf{W}^T(1)\mathbf{X}(1)$

因而 $\mathbf{W}(2) = \mathbf{W}(1) + \mathbf{0} = (0,0,0,1)^T$

② 令 $\mathbf{X}(2) = (1,0,0,1)^T, \mathbf{X}(2) \in \omega_1, \alpha_2 = \frac{1}{2}, r[\mathbf{X}(2)] = 1$

$$\mathbf{W}(3) = \mathbf{W}(2) + \alpha_2 \mathbf{X}(2) \operatorname{sgn}\{r[\mathbf{X}(2)] - \mathbf{W}^T(2)\mathbf{X}(2)\}$$

因为 $r[\mathbf{X}(2)] = \mathbf{W}^T(2)\mathbf{X}(2)$

因此 $\mathbf{W}(3) = \mathbf{W}(2) - \frac{1}{2}(1,0,0,1)^T = \left(-\frac{1}{2}, 0, 0, \frac{1}{2}\right)^T$

③ 令 $\mathbf{X}(3) = (1,0,1,1)^T, \mathbf{X}(3) \in \omega_1, r[\mathbf{X}(3)] = 1$

$$\mathbf{W}(4) = \mathbf{W}(3) + \alpha_3 \mathbf{X}(3) \operatorname{sgn}\{r[\mathbf{X}(3)] - \mathbf{W}^T(3)\mathbf{X}(3)\}$$

因为 $r[\mathbf{X}(3)] > \mathbf{W}^T(3)\mathbf{X}(3)$

因此 $\mathbf{W}(4) = \mathbf{W}(3) + \frac{1}{3}(1,0,1,1)^T = \left(-\frac{1}{6}, 0, \frac{1}{3}, \frac{5}{6}\right)^T$

按照同样的步骤把以下的样品逐个输入。当输入样品 $\mathbf{X} \in \omega_2$ 时, $r_i[\mathbf{X}(k)] = 0$, 以此确定函数 $\operatorname{sgn}\{r[\mathbf{X}(k)] - \mathbf{W}^T(k)\mathbf{X}(k)\}$ 的符号, 由此而决定修正量的方向。

一直进行到第15步, 即 $k=15, \alpha_k = \frac{1}{15}$, 得到可以令人满意的结果, 这时

$$\mathbf{W}(16) = (0.233, -0.239, -0.216, 0.619)^T$$

因此判别方程为

$$d(\mathbf{X}) = P(\omega | \mathbf{X}) - \frac{1}{2} = \mathbf{W}^T \mathbf{X} - \frac{1}{2} = 0.233x_1 - 0.239x_2 - 0.216x_3 + 0.119 = 0$$

2. 实现步骤

- ① 设各个权矢量的初始值为 0, 即 $\mathbf{W}_0(0) = \mathbf{W}_1(0) = \mathbf{W}_2(0) = \cdots = \mathbf{W}_9(0) = 0$ 。
- ② 输入第 k 次样品 $\mathbf{X}(k)$, 计算 $d_i(k) = \mathbf{W}_i^T(k) \mathbf{X}(k)$ 。
- ③ 若 $\mathbf{X}(k) \in \omega_i$, 则 $r_i[\mathbf{X}(k)] = 1$, 否则 $r_i[\mathbf{X}(k)] = 0$ 。
- ④ 计算 $\mathbf{W}_i(k+1)$, $\mathbf{W}_i(k+1) = \begin{cases} \mathbf{W}_i(k) + \alpha_k \mathbf{X}(k), & \mathbf{W}_i^T(k) \mathbf{X}(k) < r_i[\mathbf{X}(k)] \\ \mathbf{W}_i(k) - \alpha_k \mathbf{X}(k), & \mathbf{W}_i^T(k) \mathbf{X}(k) \geq r_i[\mathbf{X}(k)] \end{cases}$

其中 $\alpha_k = 1/k$ 。

- ⑤ 循环执行第②步, 直到属于 ω_i 类的所有样品都满足条件:

$$d_i(\mathbf{X}) > d_j(\mathbf{X}) \quad \forall j \neq i$$

3. 编程代码

```
%%%%%%%%%%
% 函数名称: zengliangjiaozheng()
% 参数: sample: 待识别样品特征
% 返回值: y: 待识别样品所属类别
% 函数功能: 增量校正算法
%%%%%%%%%
function y = zengliangjiaozheng( sample)

    clc;
    load templet pattern;
    w = zeros(26,10); % 初始化权值
    d = [];
    maxpos = 0;
    maxval = 0;
    r = [];
    flag = 1;
    num1 = 0;
    num2 = 0;
    f = 1;
    n = []; m = [];
    while flag
        flag = 0;
        num2 = num2 + 1;
        for j = 1:20
            for i = 1:10
                num1 = num1 + 1;
                % 初始化向量 r, 当前类别 r(i) 为 1
                r = [0 0 0 0 0 0 0 0 0 0];
```

```

r(i) = 1;
f = 1;
pattern(i).feature(26,j) = 1;% 末位置 1
for k = 1:10
    m = pattern(i).feature(:,j);
    d(k) = w(:,k)' * m;
end
% 判断是否为最大值,不是则 flag 为 1
for k = 1:10
    if k ~= i
        if d(i) <= d(k)
            flag = 1;
        end
    end
end
% 校正权值
for k = 1:10
    if r(k) > d(k)
        w(:,k) = w(:,k) + pattern(i).feature(:,j)/num1;
    else
        w(:,k) = w(:,k) - pattern(i).feature(:,j)/num1;
    end
end
end
end
if num2 > 400
    flag = 0;
end
end
sample(26) = 1;
h = [];
% 计算判别函数
for k = 1:10
    h(k) = w(:,k)' * sample';
end
[maxval,maxpos] = max(h);
y = maxpos - 1;

```

4. 效果图

采用增量校正算法自动分类识别效果如图 5-11 所示。



图 5-11 增量校正算法分类识别效果

5.6 LMSE 验证可分性

1. 理论基础

(1) LMSE 验证可分性

LMSE(Least Mean Square Error)算法是对准则函数引进最小均方误差这一条件而建立起来的。这种算法的主要特点是在训练过程中判定训练集是否线性可分,从而可对结果的收敛性做出判断。

LMSE 算法把对满足式 $\mathbf{XW} > 0$ 中 \mathbf{W} 的求解改为对满足

$$\mathbf{XW} = \mathbf{b} \quad (5-41)$$

的求解,式中 $\mathbf{b} = (b_1, b_2, \dots, b_R)^T$ 是各分量均为正值的一个矢量,故这两个公式是互相等价的。

设判决函数

$$J(\mathbf{W}, \mathbf{X}, \mathbf{b}) = \frac{1}{2} \sum_{j=1}^R (\mathbf{W}^T \mathbf{X}_j - b_j)^2 = \frac{1}{2} \|\mathbf{XW} - \mathbf{b}\|^2 \quad (5-42)$$

式中, $\|\mathbf{XW} - \mathbf{b}\|$ 为向量 $\mathbf{XW} - \mathbf{b}$ 的幅度,不难发现,当式(5-41)被满足时,判决函数 $J(\mathbf{W}, \mathbf{X}, \mathbf{b})$ 达到最小值。

由于 $J(\mathbf{W}, \mathbf{X}, \mathbf{b}) = \frac{1}{2} \|\mathbf{XW} - \mathbf{b}\|^2$, 故得到最小均方误差的条件为

$$\begin{cases} \partial J / \partial \mathbf{W} = 0 \\ \partial J / \partial \mathbf{b} = 0 \end{cases}$$

$$\text{因为 } \begin{cases} \partial J / \partial \mathbf{W} = \mathbf{X}^T (\mathbf{XW} - \mathbf{b}) \\ \partial J / \partial \mathbf{b} = -(\mathbf{XW} - \mathbf{b}) \end{cases}$$

故有

$$\begin{cases} \mathbf{X}^T (\mathbf{XW} - \mathbf{b}) = 0 \\ \mathbf{XW} - \mathbf{b} = 0 \end{cases} \quad (5-43)$$

由式(5-43)可得

$$\mathbf{W} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{b} = \mathbf{X}^\# \mathbf{b} \quad (5-44)$$

式中 $X^\#$ 称为矩阵 X 的规范逆矩阵。若 X 为 $m \times n$ 矩阵, 则 $X^\#$ 为 $n \times m$ 矩阵。

式(5-43)、式(5-44)确定了一些主要的迭代关系。对 \mathbf{b} 的前后两次迭代来说有

$$\mathbf{b}(k+1) = \mathbf{b}(k) + \delta\mathbf{b}(k) \quad (5-45)$$

因为 \mathbf{b} 应为正值, 再考虑到式(5-43)的条件, 则 \mathbf{b} 的增量 $\delta\mathbf{b}(k)$ 应为

$$\delta\mathbf{b}(k) = \begin{cases} 0, & \mathbf{XW}(k) - \mathbf{b}(k) \leq 0 \\ 2C[\mathbf{XW}(k) - \mathbf{b}(k)], & \mathbf{XW}(k) - \mathbf{b}(k) > 0 \end{cases}$$

式中, C 为一正的校正系数。上式也可写为

$$\delta\mathbf{b}(k) = C[\mathbf{XW}(k) - \mathbf{b}(k) + |\mathbf{XW}(k) - \mathbf{b}(k)|] \quad (5-46)$$

引入误差矢量 $\mathbf{E}(k)$ 为

$$\mathbf{E}(k) = \mathbf{XW}(k) - \mathbf{b}(k) \quad (5-47)$$

则式(5-46)可改写为

$$\delta\mathbf{b}(k) = C[\mathbf{E}(k) + |\mathbf{E}(k)|] \quad (5-48)$$

式中, $|\mathbf{E}(k)|$ 为 $\mathbf{E}(k)$ 的绝对值。

由式(5-44)和式(5-45)

$$\begin{aligned} \mathbf{W}(k+1) &= X^\# \mathbf{b}(k+1) \\ &= X^\# [\mathbf{b}(k) + \delta\mathbf{b}(k)] \\ &= X^\# \mathbf{b}(k) + X^\# \delta\mathbf{b}(k) \\ &= \mathbf{W}(k) + CX^\# [\mathbf{E}(k) + |\mathbf{E}(k)|] \end{aligned} \quad (5-49)$$

式(5-44)的初始条件

$$\mathbf{W}(1) = X^\# \mathbf{b}(1)$$

和 $\mathbf{b}(1) > 0$ 都是 LMSE 算法的基本内容。由此得到 LMSE 算法的迭代式:

设初值 $\mathbf{b}(1)$, 须使其每一分量都为正值, 即

$$\begin{cases} \mathbf{W}(1) = X^\# \mathbf{b}(1) \\ \mathbf{E}(k) = \mathbf{XW}(k) - \mathbf{b}(k) \\ \mathbf{W}(k+1) = \mathbf{W}(k) + CX^\# [\mathbf{E}(k) + |\mathbf{E}(k)|] = \mathbf{W}(k) + CX^\# |\mathbf{E}(k)| \\ \mathbf{b}(k+1) = \mathbf{b}(k) + C[\mathbf{E}(k) + |\mathbf{E}(k)|] \end{cases} \quad (5-50)$$

式中有

$$\begin{aligned} X^\# \mathbf{E}(k) &= X^\# [\mathbf{XW}(k) - \mathbf{b}(k)] \\ &= (X^T X)^{-1} X^T [\mathbf{XW}(k) - \mathbf{b}(k)] \\ &= (X^T X)^{-1} X^T (X X^\# - 1) \mathbf{b}(k) \\ &= (X^T X)^{-1} (X^T - X^T) \mathbf{b}(k) \\ &= 0 \end{aligned}$$

当然迭代式中也可先计算 $\mathbf{b}(k+1)$, 然后按 $\mathbf{W}(k+1) = X^\# \mathbf{b}(k+1)$ 来计算。

可以证明, 当模式类线性可分, 且校正系数 C 满足 $0 < C \leq 1$ 时, LMSE 算法收敛。证明收敛性的关键在于, 指出在极限情况下, 误差矢量 $\mathbf{E}(k) = \mathbf{XW}(k) - \mathbf{b}(k) = 0$, 因为方程组(5-50)指出, $\mathbf{b}(k)$ 是各分量均非负的矢量, 故若 $\mathbf{XW}(k) = \mathbf{b}(k)$, 则 $\mathbf{XW}(k) > 0$ 。

算法未能给出达到收敛的精确迭代次数, 故通常在每次迭代后均对 $\mathbf{XW}(k)$ 和 $\mathbf{E}(k)$ 进行检查。当 $\mathbf{XW}(k) > 0$ 或 $\mathbf{E}(k) = 0$ 则有解; 反之, 若 $\mathbf{E}(k)$ 变得非正, 则迭代应停止, 表明该模式

类是线性不可分的。

① 如果 $E(k) = 0$ 或 $XW(k) = b(k) > 0$, 有解。

② 如果 $E(k) \geq 0$, 说明 $XW(k) \geq b(k) > 0$, 有解。

③ 如果 $E(k)$ 的全部分量为负(但不是全部为零), 这表明该模式类别为线性不可分。因为若 $E(k)$ 没有一个分量为正值, 则 $b(k)$ 将不会变化, 所以不能求得解。

实际上, 对一线性不可分模式, 要达到 $E(k)$ 全部分量都为非正, 需要迭代很多次, 往往早在 $E(k)$ 全部分量都达到非正值以前, 就能看出其中有些分量向正值变化得极慢, 这时已能估计出造成线性不可分的某些样品, 应及早采取对策。

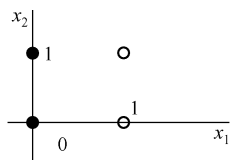


图 5-12 两类样品示意图

(2) 实例说明

下面以一个实例来说明这种情况。假设有 4 个训练样品, 共分两类, 分布如图 5-12 所示。

$$\omega_1: (0,0)^T, (0,1)^T, \omega_2: (1,0)^T, (1,1)^T$$

显然它们是线性可分的。对 ω_1, ω_2 , 所有样品增 1, 并对 ω_2 取负后, 可得样品矩阵为

$$X = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ -1 & 0 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

X 的规范逆矩阵为

$$X^\# = (X^T X)^{-1} X^T = \frac{1}{2} \begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ 3/2 & 1/2 & -1/2 & 1/2 \end{pmatrix}$$

取 $b(1) = (1,1,1,1)^T$ 和 $C=1$, 得

$$W(1) = X^\# b(1) = (-2, 0, 1)^T$$

因为

$$XW(1) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ -1 & 0 & -1 \\ -1 & -1 & -1 \end{pmatrix} \begin{pmatrix} -2 \\ 0 \\ 1 \end{pmatrix} = (1, 1, 1, 1)^T > 0$$

因而 $W(1)$ 即为所求的解, 这时

$$E(1) = XW(1) - b(1) = 0$$

现在把上述四个样品变换一下, 使出现不可分状态

$$\omega_1: (0,0)^T, (1,1)^T$$

$$\omega_2: (0,1)^T, (1,0)^T$$

这时的样品矩阵为

$$X = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & -1 & -1 \\ -1 & 0 & -1 \end{pmatrix}$$

$$\mathbf{X}^{\#} = \frac{1}{2} \begin{pmatrix} -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 3/2 & -1/2 & -1/2 & -1/2 \end{pmatrix}$$

取 $\mathbf{b}(1) = (1, 1, 1, 1)^T$ 和 $C=1$, 得

$$\mathbf{W}(1) = \mathbf{X}^{\#} \mathbf{b}(1) = (0, 0, 0)^T$$

$$\mathbf{E}(1) = \mathbf{X} \mathbf{W}(1) - \mathbf{b}(1) = (-1, -1, -1, -1)^T$$

$\mathbf{E}(1)$ 是一个全负分量的矢量, 说明系统是线性不可分的。

2. 实现步骤

实现步骤如下:

- ① 每类各选取 20 个样品, 构成样品矩阵 \mathbf{X} 。
- ② 计算 \mathbf{X} 的规范逆矩阵。
- ③ 以 $\mathbf{b}(1) = (1, 1, \dots, 1)_{1 \times 40}^T$ 和 $C=1$ 开始迭代计算 \mathbf{W} 和 \mathbf{E} 。

$$\mathbf{W}(k) = \mathbf{X}^{\#} \mathbf{b}(k), \quad \mathbf{E}(k) = \mathbf{X} \mathbf{W}(k) - \mathbf{b}(k)$$

- ④ 求出 \mathbf{E} 是一个全负分量的矢量, 说明系统是线性不可分的。
- ⑤ \mathbf{b} 按照下式变化

$$\mathbf{b}(k+1) = \mathbf{b}(k) + C[\mathbf{E}(k) + |\mathbf{E}(k)|]$$

- ⑥ 循环执行③, 对 $\mathbf{E}(k)$ 监督, 可以知道对样品的循环训练是否可以得到收敛的结果。

3. 编程代码

```
% 函数名称: ifclassified()
% 参数: class1, class2: 要验证的两个类别
% 返回值: y: 1 可分, 2 不可分
% 函数功能: 用 LMSE 方法验证两类的可分性, 返回每次迭代的权值
function y = ifclassified(class1, class2)
    load templet pattern
    clc;
    % 样品矩阵增 1
    pattern(class1).feature(26, 1:20) = 1;
    pattern(class2).feature(26, 1:20) = 1;
    % 取 class1 的前 20 个样本
    x = pattern(class1).feature(:, 1:20)';
    % 取 class2 的前 20 个样本并取负
    pattern(class2).feature(:, 1:20) = pattern(class2).feature(:, 1:20) * -1;
    x = [x; pattern(class2).feature(:, 1:20)'];
    xsharp = inv(x' * x) * x'; % 求 X 的规范逆矩阵
    b = ones(40);
    b = b(:, 1);
    flag = 1;
```

```

f=0;
fl=0;
w=[];
e=[];
c=0;
y1=0;
while flag
    fl=0;
    f=0;
    w=xsharp*b;%计算 W
    e=x*w-b;%计算 E
    c=c+1;
    str=['第' num2str(c) '次迭代的 e 值:'];
    disp(str);
    disp(e');
    for i=1:40
        if e(i)>=0&~(any(isnan(e(i))))%E 有非负并且不全是 NaN? 如果否,则不可分
            f=1;
        end
    end
    end
    if f
        for i=1:40
            if abs(e(i))>1e-10% E(i)的绝对值大于阈值? 是则调整 b 值,否则可分
                fl=1;
            end
        end
        if fl
            b=b+(e+abs(e));%调整 b 值
        else
            y1=1;
            flag=0;
        end
    else
        y1=2;
        flag=0;
    end
    if(c>1000)
        flag=0;%迭代次数超过 1000 则退出
    end
end
y=y1;

```

4. 效果图

数字 4 和数字 6 可分性验证结果如图 5-13 所示。使用 LMSE 算法验证可分性计算结果,如

图 5-14 所示,迭代到第 28 次时, e 值非全负,表示数字 4 和数字 6 是可分的。

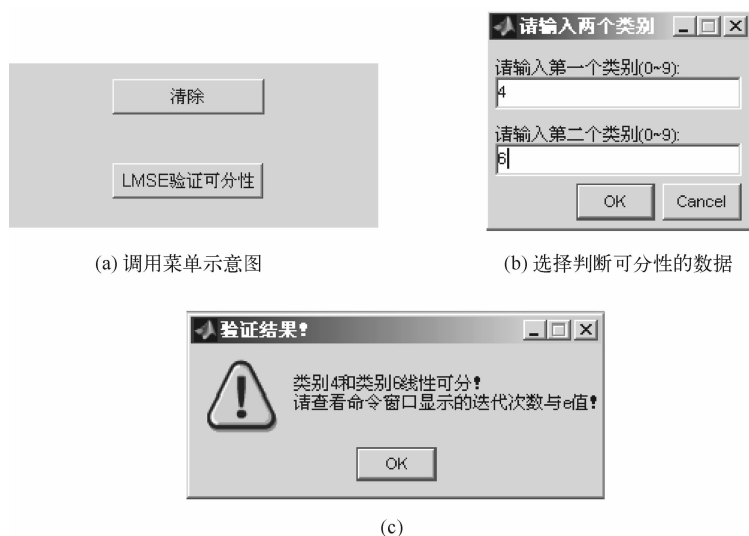


图 5-13 数字 4 和数字 6 可分性验证结果

Columns 21 through 30

0.0050 0.0088 -0.0018 0.0206 -0.0217 -0.0000 0.0412 -0.0555 -0.0308 0.0346

Columns 31 through 40

0.0044 -0.0125 0.0161 0.0019 0.0046 -0.0214 0.0173 -0.0135 0.0186 -0.0121

第28次迭代的 e 值:

1.0e-010 *

Columns 1 through 10

0.0615 0.2971 0.1357 0.0008 -0.0537 -0.5787 -0.1434 -0.2065 0.2882 0.0168

Columns 11 through 20

-0.0699 0.3970 -0.1118 0.0920 -0.1731 -0.0012 -0.1827 0.1414 0.0369 0.0001

Columns 21 through 30

0.0456 -0.1123 0.1271 0.0089 -0.1036 -0.0261 -0.1157 -0.1874 -0.0245 -0.0152

Columns 31 through 40

0.0849 0.1446 -0.0179 0.0230 0.0068 0.2129 0.0224 0.0017 -0.0708 -0.0226

图 5-14 使用 LMSE 算法验证可分性计算结果

5.7 LMSE 分类算法

1. 理论基础

(1) LMSE 分类算法

LMSE 算法以最小均方误差作为准则,因均方误差为

$$E\{[r_i(\mathbf{X}) - \mathbf{W}_i^T \mathbf{X}]^2\}$$

因而准则函数为

$$J(\mathbf{W}_i, \mathbf{X}) = \frac{1}{2} E\{[r_i(\mathbf{X}) - \mathbf{W}_i^T \mathbf{X}]^2\} \quad (5-51)$$

准则函数在 $r_i(\mathbf{X}) - \mathbf{W}_i^T \mathbf{X} = 0$ 时得 $J(\mathbf{W}_i, \mathbf{X})$ 的最小值。准则函数对 \mathbf{W}_i 的偏导数为

$$\frac{\partial J}{\partial \mathbf{W}_i} = E\{-\mathbf{X}[r_i(\mathbf{X}) - \mathbf{W}_i^T \mathbf{X}]\} \quad (5-52)$$

代入迭代方程

$$\mathbf{W}_i(k+1) = \mathbf{W}_i(k) + \alpha_k \mathbf{X}(k) \{r_i[\mathbf{X}(k)] - \mathbf{W}_i^T(k) \mathbf{X}(k)\} \quad (5-53)$$

对于多类问题来说, M 类问题应该有 M 个权函数方程, 对于每一个权函数方程来说, 如 $\mathbf{X}(k) \in \omega_i$, 则

$$r_i[\mathbf{X}(k)] = 1$$

否则

$$r_j[\mathbf{X}(k)] = 0 \quad j = 1, 2, \dots, M, j \neq i$$

(2) 实例说明

下面用一个实例来说明 LMSE 算法的进行过程。给定两类样品:

$$\omega_1: (0,0,0)^T, (1,0,0)^T, (1,0,1)^T, (1,1,0)^T$$

$$\omega_2: (0,0,1)^T, (0,1,0)^T, (0,1,1)^T, (1,1,1)^T$$

现在用 LMSE 算法计算判别函数。在进行迭代运算之前, 各样品的特征向量经过增 1:

$$\omega_1: (0,0,0,1)^T, (1,0,0,1)^T, (1,0,1,1)^T, (1,1,0,1)^T$$

$$\omega_2: (0,0,1,1)^T, (0,1,0,1)^T, (0,1,1,1)^T, (1,1,1,1)^T$$

取 $\mathbf{W}(1) = \mathbf{0}$, $\alpha_k = 1/k$ 。LMSE 算法判别函数实现步骤如下:

① 令 $\mathbf{X}(1) = (0,0,0,1)^T$, $\mathbf{X}(1) \in \omega_1$, $\alpha_1 = 1$, $r[\mathbf{X}(1)] = 1$,

因此 $\mathbf{W}(2) = \mathbf{W}(1) + \alpha_1 \mathbf{X}(1) [1 - \mathbf{W}^T(1) \mathbf{X}(1)] = (0,0,0,1)^T$

② $\mathbf{X}(2) = (1,0,0,1)^T$, $\mathbf{X}(2) \in \omega_1$, $\alpha_2 = \frac{1}{2}$, $r[\mathbf{X}(2)] = 1$

因此 $\mathbf{W}(3) = \mathbf{W}(2) + \frac{1}{2} \mathbf{X}(2) [1 - \mathbf{W}^T(2) \mathbf{X}(2)] = (0,0,0,1)^T$

③ $\mathbf{X}(3) = (1,0,1,1)^T$, $\mathbf{X}(3) \in \omega_1$, $r[\mathbf{X}(3)] = 1$

因此 $\mathbf{W}(4) = \mathbf{W}(3) = (0,0,0,1)^T$

④ $\mathbf{X}(4) = (1,1,0,1)^T$, $\mathbf{X}(4) \in \omega_1$, $r[\mathbf{X}(4)] = 1$,

因此 $\mathbf{W}(5) = \mathbf{W}(4) = (0,0,0,1)^T$

$$\textcircled{5} \mathbf{X}(5) = (0, 0, 1, 1)^T, \mathbf{X}(5) \in \omega_2, r[\mathbf{X}(5)] = 0,$$

$$\text{因此 } \mathbf{W}(6) = \mathbf{W}(5) + \alpha_5 \mathbf{X}(5) [0 - \mathbf{W}^T(5) \mathbf{X}(5)] = \left(0, 0, -\frac{1}{5}, \frac{4}{5}\right)^T$$

经过 19 次迭代以后,得到可以满足要求的结果为

$$\mathbf{W}(20) = (0.135, -0.238, -0.305, 0.721)^T$$

因此判别函数为

$$d(\mathbf{X}) = \mathbf{W}^T \mathbf{X} - 0.5 = 0.135x_1 - 0.238x_2 - 0.305x_3 + 0.221 = 0$$

2. 算法实现

- ① 设各个权矢量的初始值为 0, 即 $\mathbf{W}_0(0) = \mathbf{W}_1(0) = \mathbf{W}_2(0) = \cdots = \mathbf{W}_9(0) = 0$ 。
- ② 输入第 k 次样品 $\mathbf{X}(k)$, 计算 $d_i(k) = \mathbf{W}_i^T(k) \mathbf{X}(k)$ 。
- ③ 若 $\mathbf{X}(k) \in \omega_i$, 则 $r_i[\mathbf{X}(k)] = 1$, 否则 $r_i[\mathbf{X}(k)] = 0$ 。
- ④ 计算 $\mathbf{W}_i(k+1)$, $\mathbf{W}_i(k+1) = \mathbf{W}_i(k) + \alpha_k \mathbf{X}(k) \{r_i[\mathbf{X}(k)] - \mathbf{W}_i^T(k) \mathbf{X}(k)\}$, 其中 $\alpha_k = 1/k$ 。
- ⑤ 循环执行第②步, 直到属于 ω_i 类的所有样品都满足条件:

$$d_i(\mathbf{X}) > d_j(\mathbf{X}) \quad \forall j \neq i$$

3. 编程代码

```
%%%%%%%%%%
% 函数名称:lmseclassify()
% 参数:sample:待识别样品特征
% 返回值:y:待识别样品所属类别
% 函数功能:LMSE 算法
%%%%%%%%%%%%
function y = lmseclassify( sample)
    clc;
    load templet pattern;
    w = zeros(26,10);% 初始化权值
    flag = 1;
    num = 0;
    num1 = 0;
    d = [];
    m = [];
    r = [];
    while flag
        flag = 0;
        num1 = num1 + 1;
        for j = 1:40
            for i = 1:10
                num = num + 1;
                r = [0 0 0 0 0 0 0 0 0];
                r(i) = 1;
                pattern(i).feature(26,j) = 1;% 向量增 1
```

```

for k = 1:10
    m = pattern(i).feature(:,j);
    d(k) = w(:,k)' * m; % 计算 d
end
for k = 1:10
    if k ~= i
        if d(i) <= d(k) % d(i) 不是最大,则继续迭代
            flag = 1;
        end
    end
end
end
% 调整权值
for k = 1:10
    w(:,k) = w(:,k) + m * (r(k) - d(k)) / num;
end
end
end
if num1 > 200 % 超过迭代次数则退出
    flag = 0;
end
end
sample(26) = 1;
h = [];
for k = 1:10
    h(k) = w(:,k)' * sample'; % 计算判别函数
end
[maxval, maxpos] = max(h);
y = maxpos - 1;

```

4. 效果图

应用 LMSE 算法自动分类识别效果如图 5-15 所示。



(a) 待测样品



(b) 分类结果

图 5-15 LMSE 方法自动分类识别效果

5.8 Fisher 分类

1. 理论基础

在应用统计方法解决模式识别问题时,经常会遇到所谓的“维数灾难”的问题,在低维空间里适用的方法在高维空间里可能完全不适用。因此压缩特征空间的维数有时是很重要的。Fisher 方法实际上涉及维数压缩的问题。

如果把多维特征空间的点投影到一条直线上,就能把特征空间压缩成一维,这个在数学上是很容易办到的。但是,在高维空间里很容易分开的样品,把它们投影到任意一条直线上,有可能不同类别的样品就混在一起了,无法区分,如图 5-16(a) 所示,投影到 x_1 或 x_2 轴无法区分。若把直线绕原点转动一下,就有可能找到一个方向,样品投影到这个方向的直线上,各类样品就能很好地分开,如图 5-16(b) 所示。因此直线方向的选择是很重要的。一般来说,总能够找到一个最好的方向,使样品投影到这个方向的直线上很容易分开。如何找到这个最好的直线方向以及如何实现向最好方向投影的变换,这正是 Fisher 算法要解决的基本问题,这个投影变换恰是我们所寻求的解向量 \mathbf{W}^* 。

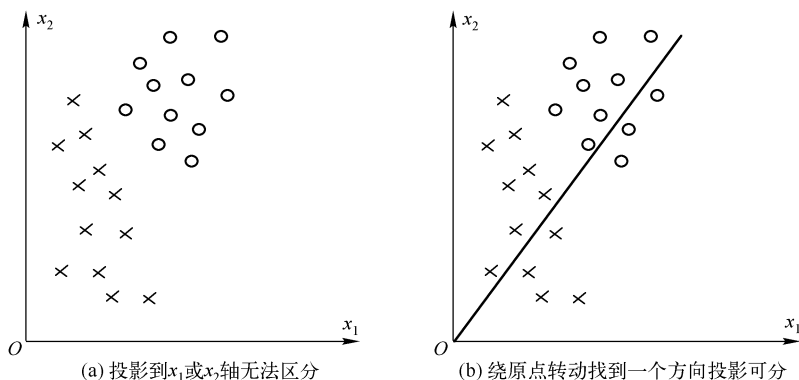


图 5-16 Fisher 线性判别原理示意图

样品训练集以及待测样品的特征总数目为 n 。为了找到最佳投影方向,需要计算出各类样品均值、样品类内离散度矩阵 \mathbf{S}_i 和总类间离散度矩阵 \mathbf{S}_w 、样品类间离散度矩阵 \mathbf{S}_b , 根据 Fisher 准则, 找到最佳投影向量, 将训练集内所有样品进行投影, 投影到一维 Y 空间, 由于 Y 空间是一维的, 则要求出 Y 空间的划分边界点, 找到边界点后, 就可以对待测样品进行一维 Y 空间的投影, 判断它的投影点与分界点的关系, 将其归类。具体方法如下:

- ① 计算各类样品均值向量 \mathbf{m}_i , \mathbf{m}_i 是各个类的均值, N_i 是 ω_i 类的样品个数。

$$\mathbf{m}_i = \frac{1}{N_i} \sum_{X \in \omega_i} \mathbf{X} \quad i = 1, 2 \quad (5-54)$$

- ② 计算样品类内离散度矩阵 \mathbf{S}_i 和总类内离散度矩阵 \mathbf{S}_w 。

$$\mathbf{S}_i = \sum_{X \in \omega_i} (\mathbf{X} - \mathbf{m}_i)(\mathbf{X} - \mathbf{m}_i)^T \quad i = 1, 2 \quad (5-55)$$

$$\mathbf{S}_w = \mathbf{S}_1 + \mathbf{S}_2 \quad (5-56)$$

③ 计算样品类间离散度矩阵 S_b 。

$$S_b = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T \quad (5-57)$$

④ 求向量 \mathbf{W}^* 。

我们希望投影后,在一维 Y 空间里各类样品尽可能地分开,也就是说我们希望两类样品均值之差 $(\tilde{\mathbf{m}}_1 - \tilde{\mathbf{m}}_2)$ 越大越好,同时希望各类样品内部尽量密集,即希望类内离散度越小越好,因此,我们可以定义 Fisher 准则函数为

$$J_F(\mathbf{W}) = \frac{\mathbf{W}^T S_b \mathbf{W}}{\mathbf{W}^T S_w \mathbf{W}} \quad (5-58)$$

使得 $J_F(\mathbf{W})$ 取得最大值的 \mathbf{W}^* 为

$$\mathbf{W}^* = S_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2) \quad (5-59)$$

⑤ 将训练集内所有样品进行投影。

$$y = (\mathbf{W}^*)^T \mathbf{X} \quad (5-60)$$

⑥ 计算在投影空间上的分割阈值 y_0 。

在一维 Y 空间,各类样品均值 $\tilde{\mathbf{m}}_i$ 为

$$\tilde{\mathbf{m}}_i = \frac{1}{N_i} \sum_{y \in w_i} y \quad i = 1, 2 \quad (5-61)$$

样品类内离散度 \tilde{s}_i^2 和总类内离散度 \tilde{s}_w 为

$$\tilde{s}_i^2 = \sum_{y \in w_i} (y - \tilde{\mathbf{m}}_i)^2 \quad i = 1, 2 \quad (5-62)$$

$$\tilde{s}_w = \tilde{s}_1^2 + \tilde{s}_2^2 \quad (5-63)$$

阈值 y_0 的选取可以有不同的方案,较常用的一种是

$$y_0 = \frac{N_1 \tilde{\mathbf{m}}_1 + N_2 \tilde{\mathbf{m}}_2}{N_1 + N_2} \quad (5-64)$$

另一种是

$$y_0 = \frac{\tilde{\mathbf{m}}_1 + \tilde{\mathbf{m}}_2}{2} + \frac{\ln(P(\omega_1)/P(\omega_2))}{N_1 + N_2 - 2} \quad (5-65)$$

⑦ 对于给定的 \mathbf{X} , 计算出它在 \mathbf{W}^* 上的投影点 y 。

$$y = (\mathbf{W}^*)^T \mathbf{X}$$

⑧ 根据决策规则分类

$$\begin{cases} y > y_0 \Rightarrow \mathbf{X} \in \omega_1 \\ y < y_0 \Rightarrow \mathbf{X} \in \omega_2 \end{cases} \quad (5-66)$$

2. 实现步骤

要实现 Fisher 分类,首先实现两类 Fisher 算法,两类 Fisher 算法能够返回最接近待测样品的类别,然后用返回的类别和新的类别做两类 Fisher 运算,又能够得到比较接近的类别,以此类推,直至所有的类别,最后得出未知样品的类别。

两类 Fisher 算法实现步骤如下:

① 求两类样品均值向量 \mathbf{m}_1 和 \mathbf{m}_2 。

② 求两类样品类内离散度矩阵 S_i 。

- ③ 求总类内离散度矩阵 S_w 。
- ④ 求向量 W^* , $W^* = S_w^{-1}(m_1 - m_2)$ 。
- ⑤ 对于两类已知样品,求出它们在 W^* 上的投影点 y_i 。
- ⑥ 求各类样品均值 \tilde{m}_i , $\tilde{m}_i = \frac{1}{N_i} \sum_{y \in w_i} y$ 。
- ⑦ 选取阈值 y_0 , 在这里取 $y_0 = \frac{N_1 \tilde{m}_1 + N_2 \tilde{m}_2}{N_1 + N_2}$ 。
- ⑧ 对于未知样品 X , 计算它在 W^* 上的投影点 y 。
- ⑨ 根据决策规则分类。

3. 算法代码

```
%%%%%%%%%%%%%%%
% 函数名称:fisher()
% 参数:sample:待识别样品特征
% 返回值: y:待识别样品所属类别
% 函数功能:Fisher 分类器
%%%%%%%%%%%%%%
function y = fisher( sample );
    clc;
    num = zeros( 1,10 );
    classnum = 0;
    for i = 1:10
        for j = 1:i
            classnum = fisherclassify( i,j, sample );
            num( classnum ) = num( classnum ) + 1;
        end
    end
    [ max_val, max_pos ] = max( num );
    y = max_pos - 1;

% -----
%%%%%%%%%%%%%%
% 函数名称:fisherclassify()
% 参数:sample:待识别样品特征 class1,class2:0~9 中的任意两个类别
% 返回值: classfit:返回与样品 sample 最接近的类别
% 函数功能:两类 Fisher 分类器
%%%%%%%%%%%%%%
function classfit = fisherclassify( class1, class2, sample )
    load templet pattern;
    % 求两类样品均值向量
    m1 = ( mean( pattern( class1 ). feature' ) )';
    m2 = ( mean( pattern( class2 ). feature' ) )';
    % 求两类样品类内离散度矩阵
```

```

s1 = cov(pattern(class1).feature') * (pattern(class1).num-1);
s2 = cov(pattern(class2).feature') * (pattern(class2).num-1);
sw = s1 + s2;%求总类间离散度矩阵
sb = (m1 - m2) * (m1 - m2)';%求样品类间离散度矩阵
w = inv(sw) * (m1 - m2);%求 w
%求已知类别在 w 上的投影
y1 = w' * pattern(class1).feature;
y2 = w' * pattern(class2).feature;
%求各类别样品在投影空间上的均值
mean1 = mean(y1');
mean2 = mean(y2');
%求阈值 y0
y0 = (pattern(class1).num * mean1 + pattern(class2).num * mean2) / ...
      (pattern(class1).num + pattern(class2).num);
%对于未知样本 sample,计算在 w 上的投影 y
y = w' * sample';
%根据决策规则分类
if y > y0
    classfit = class1;
else
    classfit = class2;
end

```

4. 效果图

应用 Fisher 算法识别效果如图 5-17 所示。



(a) 待测样品



(b) 分类结果

图 5-17 应用 Fisher 算法识别效果

5.9 基于核的 Fisher 分类

1. 理论基础

随着科学技术的迅速发展和研究对象的日益复杂,高维数据的统计分析方法显得越

来越重要。直接对高维数据进行处理会碰到很多困难,包括随着维数的增加,计算量迅速增大;数据的可视性差;维数灾难(the curse of the dimensionality),即当维数较高时,即使数据的样本点很多,散布在高维空间中的样本点仍显得很稀疏,许多在低维时应用成功的数据处理方法,在高维中不能应用,如关于密度函数估计的核估计法,邻域法等;低维时鲁棒性很好的统计方法到了高维,其稳健性也就变差了。由此可见在多元统计过程中降维的重要性。

然而,一般常见的降维方法是建立在正态分布这一假设基础上的线性方法,显得过于简化,因而往往不能满足现实中的需要。这里将传统的线性降维方法通过引入核函数推广到非线性领域中。

基于核方法的特征提取理论本质上都是基于样本的,因此,它不仅适合解决非线性特征提取问题,而且还能比线性降维方法提供更多的特征数目和更好的特征质量,因为前者可提供的特征数目与输入样本的数目是相等的,而后的特征数目仅为输入样本的维数。

核方法首先采用非线性映射将原始数据由数据空间映射到特征空间,进而在特征空间进行对应的线性操作。由于运用了非线性映射,且这种非线性映射往往是非常复杂的,从而大大增强了非线性数据处理能力。

从本质上讲,核方法实现了数据空间、特征空间和类别空间之间的非线性变换。设 \mathbf{X}_i 和 \mathbf{X}_j 为数据空间中的样本点,数据空间到特征空间的映射函数为 Φ ,核方法的基础是实现向量的内积变换 $\langle \mathbf{X}_i, \mathbf{X}_j \rangle \rightarrow k(\mathbf{X}_i, \mathbf{X}_j) = \Phi(\mathbf{X}_i) \cdot \Phi(\mathbf{X}_j)$ 。

通常,非线性变换函数 $\Phi(\cdot)$ 相当复杂,而运算过程中实际用到的核函数 $k(\cdot, \cdot)$ 则相对简单得多,这也正是核方法最为迷人的地方。

(1) 几种常用核函数形式

核函数必须满足 Mercer 条件。目前,获得应用的核函数有以下几种形式:

① 线性核函数

$$k(\mathbf{X}, \mathbf{Y}) = \langle \mathbf{X}, \mathbf{Y} \rangle \quad (5-67)$$

② 二次核函数

$$k(\mathbf{X}, \mathbf{Y}) = \langle \mathbf{X} \cdot \mathbf{Y} \rangle (\langle \mathbf{X} \cdot \mathbf{Y} \rangle + 1) \quad (5-68)$$

③ 多项式核函数。多项式是最常使用的一种非线性映射, d 阶的多项式核函数定义如下

$$k(\mathbf{X}, \mathbf{Y}) = (\langle \mathbf{X} \cdot \mathbf{Y} \rangle + c)^d \quad (5-69)$$

式中, c 为常数, d 为多项式阶数。当 $c=0, d=1$ 时,该核函数即为线性核函数。

④ 高斯径向基(RBF)函数。最通用的径向基函数采用高斯径向基函数,定义为

$$k(\mathbf{X}, \mathbf{Y}) = \exp\left\{-\frac{|\mathbf{X} - \mathbf{Y}|^2}{2\sigma^2}\right\} \quad (5-70)$$

式中, $|\mathbf{X} - \mathbf{Y}|$ 为两个向量之间的距离, σ 为常数。

⑤ 多层感知器核函数(又称 Sigmoid 核函数)

$$k(\mathbf{X}, \mathbf{Y}) = \tanh(\text{scale} \times \langle \mathbf{X} \cdot \mathbf{Y} \rangle - \text{offset}) \quad (5-71)$$

式中, scale 和 offset 是尺度和衰减参数。

实际上,在核方法的应用中,核函数的选择及相关参数的确定是问题的关键和难点所在,到目前为止,也没有太多的理论指导。需要指出的是,与同一个核函数对应的映射可能不唯

一。在实际应用中,甚至使用者无需知道映射函数的具体形式。与同一个核函数对应的不同映射在维数上也存在差别。

(2) 输入数据空间到特征空间的非线性映射 $\Phi: X \rightarrow F$

在特征空间中往往存在输入空间所没有的性能,考虑输入空间中两类线性不可分样本 $X = (x_1, x_2)^T$, 一类为 $\{(1, 0), (0, 1)\}$, 另一类为 $\{(0, 0), (1, 1)\}$; 而与式 $\Phi(x) = (x_1 \cdot x_1, \sqrt{2} \cdot x_1 \cdot x_2, x_2 \cdot x_2)$ 对应的特征空间中相应的一类为 $\{(1, 0, 0), (0, 0, 1)\}$, 另一类为 $\{(0, 0, 0), (1, \sqrt{2}, 1)\}$, 变为线性可分。

从上面的讨论可见,核方法可以将线性空间中的非线性问题映射为非线性空间中的线性问题,从而有望较好地克服以往线性方法处理非线性问题所存在的不足。

Fisher 判别函数过于简单,往往不能满足处理非线性数据的要求。改进的途径有两条:一是对样本集进行复杂的概率密度估计,在此基础上再使用贝叶斯最优分类器,这种方法在理论上是最理想的,然而由于需要极多的数据样本,在实际中常常是不可行的;第二条途径是采用非线性投影,使投影后的数据线性可分。核 Fisher 判别分析使用了类似于 SVM 和核 PCA 方法的“核技巧”,即首先把数据非线性地映射到某个特征空间 F , 然后在这个特征空间中进行 Fisher 线性判别,这样就隐含地实现了原输入空间的非线性判别。

设 Φ 是输入空间到某个特征空间 F 的非线性映射: $\Phi: X \rightarrow F$ 。

通过非线性映射,输入空间中的向量集合 X_1, X_2, \dots, X_N 映射为特征空间中的向量集合

$$\Phi(X_1), \Phi(X_2), \dots, \Phi(X_N)$$

则在特征空间中可定义两类样本的均值向量 m_i^Φ 为

$$m_i^\Phi = \left(\frac{1}{N_i} \right) \sum_{X \in X_i} \Phi(X) \quad (i = 1, 2) \quad (5-72)$$

样本类间离散度矩阵 S_b^Φ 为

$$S_b^\Phi = (m_1^\Phi - m_2^\Phi)(m_1^\Phi - m_2^\Phi)^T \quad (5-73)$$

总类内离散度矩阵 S_w^Φ 为

$$S_w^\Phi = \sum_{i=1,2} \sum_{X \in X_i} (\Phi(X) - m_i^\Phi)(\Phi(X) - m_i^\Phi)^T \quad (5-74)$$

(3) 特征空间中 $\Phi(X)$ 在 W 上的投影变换

设投影直线的方向为 W , 则投影后应有

$$\max J_F(W) = \frac{W^T S_b^\Phi W}{W^T S_w^\Phi W} \quad (5-75)$$

由式(5-75)解得的最优投影方向为

$$W^* = (S_w^\Phi)^{-1} (m_1^\Phi - m_2^\Phi) \quad (5-76)$$

$\Phi(X)$ 在 W 上的投影为

$$y = W^{*T} \Phi(X) \quad (5-77)$$

考虑到 W 可由 $\Phi(X_1), \Phi(X_2), \dots, \Phi(X_N)$ 线性表示, 即有

$$W = \sum_{i=1}^N \alpha_i \Phi(X_i) \quad (5-78)$$

结合式(5-72)和式(5-78),有

$$\bar{y}_i = \mathbf{W}^T \mathbf{m}_i^\Phi = \frac{1}{N_i} \sum_{j=1}^N \sum_{k=1}^{N_i} \alpha_j k(\mathbf{X}_j, \mathbf{X}_k^{(\omega_i)}) = \alpha^T \mathbf{M}_i \quad (i = 1, 2; j = 1, 2, \dots, N) \quad (5-79)$$

式中定义 \mathbf{M}_i 为一 $N \times 1$ 的矩阵,且

$$(\mathbf{M}_i)_j = \left(\frac{1}{N_i} \right) \sum_{k=1}^{N_i} k(\mathbf{X}_j, \mathbf{X}_k^{(\omega_i)}) \quad (i = 1, 2; j = 1, 2, \dots, N) \quad (5-80)$$

结合式(5-73)和式(5-79)有

$$\mathbf{W}^T \mathbf{S}_b^\Phi \mathbf{W} = \mathbf{W}^T (\mathbf{m}_1^\Phi - \mathbf{m}_2^\Phi) (\mathbf{m}_1^\Phi - \mathbf{m}_2^\Phi)^T \mathbf{W} = \alpha^T (\mathbf{M}_1 - \mathbf{M}_2) (\mathbf{M}_1 - \mathbf{M}_2)^T \alpha = \alpha^T \mathbf{M} \alpha \quad (5-81)$$

上式中定义

$$\mathbf{M} = (\mathbf{M}_1 - \mathbf{M}_2) (\mathbf{M}_1 - \mathbf{M}_2)^T \quad (5-82)$$

结合式(5-74)和式(5-78)有

$$\mathbf{W}^T \mathbf{S}_w^\Phi \mathbf{W} = \mathbf{W}^T \sum_{i=1,2} \sum_{\mathbf{x} \in X_i} (\Phi(\mathbf{x}) - \mathbf{m}_i^\Phi) (\Phi(\mathbf{x}) - \mathbf{m}_i^\Phi)^T \mathbf{W} = \alpha^T \mathbf{H} \alpha \quad (5-83)$$

式中

$$\mathbf{H} = \sum_{i=1,2} \mathbf{K}_i (\mathbf{I} - \mathbf{L}_i) \mathbf{K}_i^T \quad (5-84)$$

式中, \mathbf{K}_i 为 $N \times N_i$ ($i=1,2$) 矩阵,并满足

$$(\mathbf{K}_i)_{p,q} = k(\mathbf{X}_p, \mathbf{X}_q^{(\omega_i)}) \quad (p = 1, 2, \dots, N; q = 1, 2, \dots, N_i) \quad (5-85)$$

即 \mathbf{K}_i 为第 i 类的核矩阵, \mathbf{I} 为 $N_i \times N_i$ 大小的单位阵; \mathbf{L}_i 为 $N_i \times N_i$ 大小的矩阵,其所有元素都为 $1/N_i$ 。

则结合式(5-81)和式(5-83),式(5-75)等价为

$$\max J(\alpha) = \frac{\alpha^T \mathbf{M} \alpha}{\alpha^T \mathbf{H} \alpha} \quad (5-86)$$

可见, α 实质是矩阵 $\mathbf{H}^{-1} \mathbf{M}$ 的最大特征值对应的特征向量。 α 可以直接如下求得

$$\alpha = \mathbf{H}^{-1} (\mathbf{M}_1 - \mathbf{M}_2) \quad (5-87)$$

为了求解 \mathbf{W} , 需要使 \mathbf{H} 为正定的。为此, 我们简单地对矩阵 \mathbf{H} 加上一个量 μ , 即用 \mathbf{H}_μ 替代 \mathbf{H} , 即

$$\mathbf{H}_\mu = \mathbf{H} + \mu \mathbf{I} \quad (5-88)$$

式中, \mathbf{I} 为单位阵。

特征空间中 $\Phi(\mathbf{X})$ 在 \mathbf{W} 上的投影变换为 $k(\cdot, \mathbf{X})$ 在 α 上的投影, 即

$$y = \mathbf{W}^T \cdot \Phi(\mathbf{X}) = \sum_{j=1}^N \alpha_j k(\mathbf{X}_j, \mathbf{X}) \quad (5-89)$$

(4) 确定分界阈值点 y_0

① $\tilde{\mathbf{m}}_i^\Phi$ ($i=1,2$) 为投影后的各类别的平均值, $\tilde{\mathbf{m}}_i^\Phi$ 满足

$$\tilde{\mathbf{m}}_i^\Phi = \frac{1}{N_i} \sum_{\mathbf{x}_j \in \omega_i} y_j = \frac{1}{N_i} \sum_{\mathbf{x} \in \omega_i} \mathbf{W}^T \Phi(\mathbf{x}) = \frac{1}{N_i} \sum_{\mathbf{x} \in \omega_i} \sum_{j=1}^N \alpha_j k(\mathbf{X}_j, \mathbf{x}) \quad (5-90)$$

② 对于核 Fisher 线性判别法, 分界阈值点 y_0 可选为

$$y_0 = \frac{N_1 \tilde{\mathbf{m}}_1^\Phi + N_2 \tilde{\mathbf{m}}_2^\Phi}{N_1 + N_2} \quad (5-91)$$

(5) 待测样品投影

将待测样品也进行投影得到 y 。 $y > y_0$, 则 x 属于 ω_1 。 $y < y_0$, 则 x 属于 ω_2 。

2. 实现步骤

同 Fisher 算法, 基于核的 Fisher 分类同样要首先实现两类分类, 返回最接近待测样品的类别, 然后用返回的类别和新的类别做两类分类, 又能够得到比较接近的类别, 以此类推, 最后得出未知样品的类别。

两类基于核的 Fisher 算法步骤如下:

- ① 求 $(\mathbf{M}_i)_j = \left(\frac{1}{N_i} \right) \sum_{k=1}^{N_i} k(\mathbf{X}_j, \mathbf{X}_k^{(\omega_i)}) (i = 1, 2; j = 1, 2, \dots, N)$ 。取高斯径向基核函数 $k(\mathbf{X}_j, \mathbf{X}_k^{(\omega_i)}) = \exp \left\{ -\frac{|\mathbf{X}_j - \mathbf{X}_k^{(\omega_i)}|^2}{2\sigma^2} \right\}$
- ② 求 $\mathbf{H} = \sum_{i=1,2} \mathbf{K}_i (\mathbf{I} - \mathbf{L}_i) \mathbf{K}_i^T$ 及 $\mathbf{H}_\mu = \mathbf{H} + \mu \mathbf{I}$, 其中 \mathbf{K}_i 见式(5-85) 计算。
- ③ 求 $\boldsymbol{\alpha} = \mathbf{H}_\mu^{-1} (\mathbf{M}_1 - \mathbf{M}_2)$ 。
- ④ 求训练集内各类样品投影。 $y_j = \mathbf{W}^T \cdot \Phi(\mathbf{X}_j) = \sum_{i=1}^N \alpha_i k(\mathbf{X}_i, \mathbf{X}_j), j = 1, 2, \dots, N$ 。
- ⑤ 求均值 $\tilde{m}_i^\Phi = \frac{1}{N_i} \sum_{y_j \in \omega_i} y_j$ 。
- ⑥ 求阈值点 y_0 。
- ⑦ 对于特定样本 \mathbf{X} , 求它的投影点 y 。
- ⑧ 根据决策规则分类。

3. 编程代码

```
% 函数名称: kenfisher( )
% 参数: sample: 待识别样品特征
% 返回值: y: 待识别样品所属类别
% 函数功能: 基于核的 Fisher 分类器
function y = kenfisher( sample );
    clc;
    num = zeros( 1, 10 );
    classnum = 0;
    for i = 1:10
        for j = 1:i-1
            classnum = fisherclassify( i, j, sample );
            num( classnum ) = num( classnum ) + 1;
        end
    end
end
```

```

[ max_val, max_pos ] = max( num );
y = max_pos - 1;
% -----
% 函数名称:fisherclassify()
% 参数:sample:待识别样品特征 class1,class2:0~9 中的任意两个类别
% 返回值: classfit:返回与样品 sample 最接近的类别
% 函数功能:两类基于核的 Fisher 分类器
function classfit = fisherclassify( class1,class2,sample)

load templet pattern;
num1 = 50;% class1 类选取样品个数
num2 = 50;% class2 类选取样品个数
m1 = [];m2 = [];
l = num1 + num2;% 样品总个数
% 将两类样品合在一起存放在矩阵 p 中
p = [ pattern( class1 ). feature( :,1:num1 ) pattern( class2 ). feature( :,1:num2 ) ];
% 求 M1
for i = 1:l
    m1(i) = 0;
    for j = 1:num1
        m1(i) = m1(i) + k( p( :,i ), pattern( class1 ). feature( :,j ) );
    end
    m1(i) = m1(i)/num1;
end
m1 = m1';
% 求 M2
for i = 1:l
    m2(i) = 0;
    for j = 1:num2
        m2(i) = m2(i) + k( p( :,i ), pattern( class2 ). feature( :,j ) );
    end
    m2(i) = m2(i)/num2;
end
m2 = m2';
m = ( m1 - m2 ) * ( m1 - m2 )';% 求 M
k1 = [];k2 = [];
% 求 K1
for i = 1:l
    for j = 1:num1
        k1(i,j) = k( p( :,i ), pattern( class1 ). feature( :,j ) );
    end
end

```



```

% 求 K2
for i = 1:l
    for j = 1:num2
        k2(i,j) = k(p(:,i), pattern(class2).feature(:,j));
    end
end
% 求 N
I11 = eye(num1);
I22 = eye(num2);
I1 = ones(num1)/num1;
I2 = ones(num2)/num2;
n = k1 * (I11 - I1) * k1' + k2 * (I22 - I2) * k2';
n = n + 0.001 * eye(1); % 使 N 正定
a = inv(n) * (m1 - m2); % 求 a
y1 = []; y2 = [];
% 求两类样本的线性投影
for i = 1:num1
    y1(i) = 0;
    for j = 1:l
        y1(i) = y1(i) + a(j) * k(p(:,j), pattern(class1).feature(:,i));
    end
end
for i = 1:num2
    y2(i) = 0;
    for j = 1:l
        y2(i) = y2(i) + a(j) * k(p(:,j), pattern(class2).feature(:,i));
    end
end
% 求各类别在特征空间的均值
mean1 = mean(y1');
mean2 = mean(y2');
% 求阈值
y0 = (num1 * mean1 + num2 * mean2) / (num1 + num2);
% 对于未知样本 sample, 计算投影点
y = 0;
for i = 1:l
    y = y + a(i) * k(p(:,i), sample');
end
% 决策分类
if y > y0
    classfit = class1;
else
    classfit = class2;
end

```

```

end
% -----
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 函数名称:k()
% 参数:x1,x2:两个样本向量
% 返回值:y:核值
% 函数功能:高斯核函数
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = k(x1,x2)
    y = exp(-(x1-x2)'*(x1-x2));

```

4. 效果图

采用基于核的 Fisher 算法自动分类识别效果如图 5-18 所示。



(a) 待测样品



(b) 分类结果

图 5-18 采用基于核的 Fisher 算法自动分类识别效果

5.10 势函数法

1. 理论基础

势函数法是非线性分类器中常用到的一种方法,它借用电场的概念,来解决模式的分类问题。在势函数法中,把属于一类的样品看做正电荷,而属于另一类的样品看做负电荷,从而把模式的分类转变为正负电荷的转移,电位为 0 的等位线即为判别界线。

势函数在选择时应同时满足以下三个条件:

$K(\mathbf{X}_k, \mathbf{X}) = K(\mathbf{X}, \mathbf{X}_k)$, 当且仅当 $\mathbf{X} = \mathbf{X}_k$ 时,达到最大值。

当向量 \mathbf{X} 与 \mathbf{X}_k 的距离趋于无穷时, $K(\mathbf{X}, \mathbf{X}_k)$ 趋于 0。

$K(\mathbf{X}_k, \mathbf{X})$ 是光滑函数,且是 \mathbf{X}_k 与 \mathbf{X} 之间距离的单调减小函数。

通常选择的函数有

$$K(\mathbf{X}, \mathbf{X}_k) = \exp(-\alpha \|\mathbf{X} - \mathbf{X}_k\|^2) \quad (5-92)$$

$$K(\mathbf{X}, \mathbf{X}_k) = \frac{1}{1 + \alpha \|\mathbf{X} - \mathbf{X}_k\|^2} \quad (5-93)$$

$$K(\mathbf{X}, \mathbf{X}_k) = \left| \frac{\sin \alpha \|\mathbf{X} - \mathbf{X}_k\|^2}{\alpha \|\mathbf{X} - \mathbf{X}_k\|^2} \right| \quad (5-94)$$

(1) 势函数迭代算法

势函数算法的训练过程,是利用势函数在逐个样品输入时逐步积累电位的过程。对于两类问题来说,例如,势积累方程能以其运算结果的正负来区别两类样品,则训练结束。

算法过程:

设初始电位为 $K_0(\mathbf{X}) = 0$ 。

第1步,输入样品 \mathbf{X}_1 ,计算其积累电位 $K_1(\mathbf{X})$,

$$K_1(\mathbf{X}) = \begin{cases} K_0(\mathbf{X}) + K(\mathbf{X}, \mathbf{X}_1), & \mathbf{X}_1 \in \omega_1 \\ K_0(\mathbf{X}) - K(\mathbf{X}, \mathbf{X}_1), & \mathbf{X}_2 \in \omega_2 \end{cases}$$

$K_1(\mathbf{X})$ 描述了加入第一个样品后的边界划分,样品属于 ω_1 ,势函数为正;样品属于 ω_2 ,势函数为负。

第2步,输入第二个样品 \mathbf{X}_2 ,有三种情况:

① 若 $\mathbf{X}_2 \in \omega_1$,且 $K_1(\mathbf{X}_2) > 0$ 或 $\mathbf{X}_2 \in \omega_2$,且 $K_1(\mathbf{X}_2) < 0$,表示分类正确,则势函数不变,即

$$K_2(\mathbf{X}) = K_1(\mathbf{X})$$

② 若 $\mathbf{X}_2 \in \omega_1$,但 $K_1(\mathbf{X}_2) \leq 0$,则修改势函数,令

$$K_2(\mathbf{X}) = K_1(\mathbf{X}) + K(\mathbf{X}, \mathbf{X}_2)$$

③ 若 $\mathbf{X}_2 \in \omega_2$,但 $K_1(\mathbf{X}_2) \geq 0$,应修改势函数,令

$$K_2(\mathbf{X}) = K_1(\mathbf{X}) - K(\mathbf{X}, \mathbf{X}_2)$$

可以看出:以上②、③两种情况属于错分,即若 \mathbf{X}_2 处于 $K_1(\mathbf{X})$ 所定义的边界的错误一边,则当 $\mathbf{X}_2 \in \omega_1$ 时,积累势函数 $K_2(\mathbf{X})$ 要加上 $K(\mathbf{X}, \mathbf{X}_2)$ 。反之,当 $\mathbf{X}_2 \in \omega_2$ 时,积累势函数要减去 $K(\mathbf{X}, \mathbf{X}_2)$ 。

第 $i+1$ 步,这时已输入 i 个训练样品 $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_i$,这时的积累势函数也会有三种情形:

① 若 $\mathbf{X}_{i+1} \in \omega_1$,且 $K_i(\mathbf{X}_{i+1}) > 0$,或 $\mathbf{X}_{i+1} \in \omega_2$,且 $K_i(\mathbf{X}_{i+1}) < 0$
则

$$K_{i+1}(\mathbf{X}) = K_i(\mathbf{X}) \quad (5-95)$$

② 若 $\mathbf{X}_{i+1} \in \omega_1$,且 $K_i(\mathbf{X}_{i+1}) \leq 0$ 则

$$K_{i+1}(\mathbf{X}) = K_i(\mathbf{X}) + K(\mathbf{X}, \mathbf{X}_{i+1}) \quad (5-96)$$

③ 若 $\mathbf{X}_{i+1} \in \omega_2$,且 $K_i(\mathbf{X}_{i+1}) \geq 0$ 则

$$K_{i+1}(\mathbf{X}) = K_i(\mathbf{X}) - K(\mathbf{X}, \mathbf{X}_{i+1}) \quad (5-97)$$

这三种情况可以归纳为一个方程

$$K_{i+1}(\mathbf{X}) = K_i(\mathbf{X}) + r_{i+1} K(\mathbf{X}, \mathbf{X}_{i+1}) \quad (5-98)$$

式中 r_{i+1} 的取值如表 5-2 所示。

表 5-2 势函数算法系数 r_{i+1} 的取值

X_{i+1} 类别	$K_i(X_{i+1})$	r_{i+1}
ω_1	>0	0
ω_2	<0	0
ω_1	≤ 0	1
ω_2	≥ 0	-1

如果从所给的训练样品集 $\{X_1, X_2, \dots, X_i, \dots\}$ 中省略那些并不使积累势函数发生变化的样品, 则可得一简化的样品序列 $\{X_1, X_2, \dots, X_j, \dots\}$, 它们完全是校正错误的模式样品, 由式(5-96) 和式(5-97)可以归纳为:

$$K_{i+1}(X) = \sum_{X_j} \alpha_j K(X, X_j) \quad (5-99)$$

式中

$$\alpha_j = \begin{cases} 1, & X_j \in \omega_1 \\ -1, & X_j \in \omega_2 \end{cases}$$

即由 $i+1$ 个样品产生的积累势函数, 等于 ω_1 类和 ω_2 类两者中的校正错误样品的总位势之差。

由此算法可以看出, 积累势函数不必做任何修改就可用做判别函数。设有一个两类问题, 取 $d(X) = K(X)$, 则由式(5-96)可得

$$d_{i+1}(X) = d_i(X) + r_{i+1}K(X, X_{i+1}) \quad (5-100)$$

式中系数取值由表 5-3 决定。

(2) 实例说明

设有 4 个样品, 共分两类, 分布如图 5-19 所示。

$$\omega_1: (0,0)^T, (2,0)^T$$

$$\omega_2: (1,1)^T, (1,-1)^T$$

显然这两类样品不是线性可分的。势函数取

$$K(X, X_k) = \exp \{ -\alpha [(x_1 - x_{k1})^2 + (x_2 - x_{k2})^2] \}$$

取 $\alpha = 1$, 势函数实现分类方法如下:

① 输入 $X_1 = (0,0)^T, X_1 \in \omega_1$,

$$K_1(X) = K(X, X_1) = \exp \{ -(x_1^2 + x_2^2) \}$$

② 输入 $X_2 = (2,0)^T, X_2 \in \omega_1, K_1(X_2) = e^{-4} > 0$, 分类正确, 不需修正

$$K_2(X) = K_1(X) = K(X, X_1) = \exp \{ -(x_1^2 + x_2^2) \}$$

③ 输入 $X_3 = (1,1)^T, X_3 \in \omega_2, K_2(X_3) = e^{-2} < 0$, 分类错误, 需修正

$$K_3(X) = K_2(X) - K(X, X_3) = \exp \{ -(x_1^2 + x_2^2) \} - \exp \{ -[(x_1 - 1)^2 + (x_2 - 1)^2] \}$$

④ 输入 $X_4 = (1,-1)^T, X_4 \in \omega_2, K_3(X_4) = e^{-2} - e^{-4} < 0$

$$K_4(X) = K_3(X) - K(X, X_4) = \exp \{ -(x_1^2 + x_2^2) \} - \exp \{ -[(x_1 - 1)^2 + (x_2 -$$

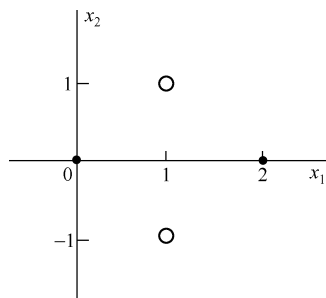


图 5-19 样品示意图

$$1)^2\} - \exp\{-[(x_1 - 1)^2 + (x_2 + 1)^2]\}$$

⑤ 再次输入 $X_1 = (0, 0)^T$, $X_1 \in \omega_1$, $K_4(X_1) = e^0 - e^{-2} - e^{-2} > 0$ 不需修正

$$K_5(X) = K_4(X)$$

⑥ 再次输入 $X_2 = (2, 0)^T$, $X_2 \in \omega_1$, $K_5(X_2) = e^{-4} - 2e^{-2} \not> 0$

$$K_6(X) = K_5(X) + K(X, X_2) = \exp\{-(x_1^2 + x_2^2)\} - \exp\{-(x_1 - 1)^2 + (x_2 - 1)^2\} - \exp\{-(x_1 - 1)^2 + (x_2 + 1)^2\} + \exp\{-(x_1 - 2)^2 + x_2^2\}$$

⑦ 再次输入 $X_3 = (1, 1)^T$, $X_3 \in \omega_2$, $K_6(X_3) = 2e^{-2} - 1 - e^{-4} < 0$, 不需修正

$$K_7(X) = K_6(X)$$

⑧ 再次输入 $X_4 = (1, -1)^T$, $X_4 \in \omega_2$, $K_7(X_4) = e^{-2} - e^{-2} - 1 + e^{-4} < 0$, 不需修正

$$K_8(X) = K_7(X)$$

⑨ 再次输入 $X_1 = (0, 0)^T$, $X_1 \in \omega_1$, $K_8(X_1) = e^0 - e^{-2} - e^{-2} + e^{-4} > 0$, 不需修正

$$K_9(X) = K_8(X)$$

⑩ 再次输入 $X_2 = (2, 0)^T$, $X_2 \in \omega_1$, $K_9(X_2) = e^{-4} - e^{-2} - e^{-2} + e^0 > 0$, 不需修正

$$K_{10}(X) = K_9(X)$$

至此为止,所有的训练样品都能被正确分类,可以得到非线性判别函数为:

$$d(X) = K_{10}(X) = \exp\{-(x_1^2 + x_2^2)\} - \exp\{-(x_1 - 1)^2 + (x_2 - 1)^2\} - \exp\{-(x_1 - 1)^2 + (x_2 + 1)^2\} + \exp\{-(x_1 - 2)^2 + x_2^2\}$$

由势函数的迭代公式可知,它具有很强的分类能力,但当修正次数增多的话,势函数方程的项数增多,使计算机的计算量大增。

2. 实现步骤

① 每一类取一个判别函数 $h_x()$, 10 个类别共有 10 个判别函数。

② 输入第一个样品 X_1 , 10 个判别函数初始值都为 $K(X, X_1)$ 。

③ 输入第 i 个样品 X_i , 若 $X_i \in \omega_j$, 判断 $h_x(j)$ 是否大于 0, 若 $h_x(j)$ 大于 0, 则 ω_j 类的判别函数不需调整, 否则按照规则 $K_i(X) = K_{i-1}(X) + K(X, X_i)$ 调整。判断 $h_x(k)$ ($k = 0, 1, 2, \dots, 9, k \neq j$) 是否小于 0, 若 $h_x(k)$ 小于 0 则 ω_k 类的判别函数不需调整, 否则按照规则 $K_i(X) = K_{i-1}(X) - K(X, X_i)$ 调整。

④ 循环第③步, 直至输入所有的样品判别函数都不需要调整。此时得到每一类的判别函数。

⑤ 输入待测样品, 判别函数的最大值的类别就是待测样品的类别。

3. 编程代码

```
%%%%%%%%%%%%%%%
% 函数名称: shihanshu()
% 参数: sample: 待识别样品特征
% 返回值: y: 待识别样品所属类别
% 函数功能: 势函数分类法
%%%%%%%%%%%%%%
function y = shihanshu( sample)
    clc;
    load templet pattern;
```

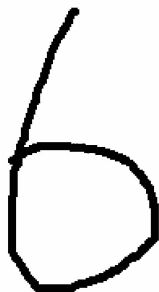
```

r = [];
x = [];
c = 0;
t = 0;
hx = [];
for i = 1:20% 每类取 20 个样本
    for n = 1:10% 十个类别
        if c == 0
            r(1:10,1) = 1;% 初始化系数矩阵
        else
            for j = 1:10
                t = 0;
                for l = 1:c
                    m = k( pattern( n). feature( :,i),pattern( x(1,1)). feature. ...
                        ( :,x(1,2)) );
                    t = t + r(j,l) * m;% 计算判别函数
                end
                hx(j) = t;
            end
            % 调整系数
            for j = 1:10
                if(j == n)
                    if hx(j) > 0% 分类正确
                        r(j,c + 1) = 0;
                    else% 分类错误
                        r(j,c + 1) = 1;
                    end
                else
                    if hx(j) < 0% 分类正确
                        r(j,c + 1) = 0;
                    else% 分类错误
                        r(j,c + 1) = -1;
                    end
                end
            end
        end
        x(c + 1,1) = n;% 类号
        x(c + 1,2) = i;% 样品号
        c = c + 1;
    end
end
for j = 1:10
    t = 0;
    for l = 1:200
        t = t + r(j,l) * k( sample',pattern( x(1,1)). feature( :,x(1,2)) );
    end
    hx(j) = t;% 代入待测样本,计算判别函数值
end

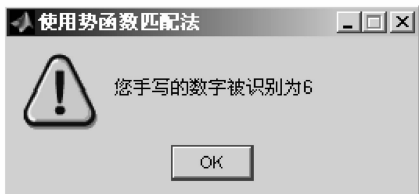
```

4. 效果图

采用势函数算法分类效果如图 5-20 所示。



(a) 待測樣品



(b) 分类结果

图 5-20 采用势函数算法分类效果

5.11 支持向量机

1. 理论基础

支持向量机(Support Vector Machine)是 Vapnik 等人根据统计学理论提出的一种新的通用学习方法,它是建立在统计学理论的 VC 维理论和结构风险最小原理基础上的,能较好地解决小样本、非线性、高维数和局部极小点等实际问题,已成为机器学习界的研究热点之一,并成功地应用于分类、函数逼近和时间序列预测等方面。

(1) 线性最优分类超平面

支持向量机的研究最初是针对模式识别中的两类线性可分问题的。设线性可分样本集, $(\mathbf{X}_i, y_i) (i=1, 2, \dots, N, \mathbf{X} \in R^n, y \in \{-1, 1\})$, 根据类别 y 不同分为正样本子集 \mathbf{X}^+ 和负样本子集 \mathbf{X}^- , 这两个子集对于超平面可分的条件是, 存在一个单位向量 $\phi (\|\phi\|=1)$ 和常数 c , 使式(5-101)成立, 其中 \cdot 是向量内积运算。

$$\begin{cases} \langle \mathbf{X}^+ \cdot \boldsymbol{\phi} \rangle > c \\ \langle \mathbf{X}^- \cdot \boldsymbol{\phi} \rangle < c \end{cases} \quad (5-101)$$

对于任何单位向量 ϕ , 确定两个值

$$\begin{cases} c_1(\phi) = \min \langle X^+ \cdot \phi \rangle \\ c_2(\phi) = \max \langle X^- \cdot \phi \rangle \end{cases} \quad (5-102)$$

找到一个 ϕ_0 , 使式(5-103)最大化

$$\gamma(\phi) = \frac{c_1(\phi) - c_2(\phi)}{2}, \quad \|\phi\| = 1 \quad (5-103)$$

由约束式(5-104)和最大化函数式(5-103)得到向量 ϕ_0 与常数 c_0 ,

$$c_0 = \frac{c_1(\phi_0) + c_2(\phi_0)}{2} \quad (5-104)$$

确定一个超平面, 将两类样本集分开, 并具有最大间隔, 参见式(5-103), 这样的超平面为最大间隔超平面, 也称最优分类超平面, 如图5-21所示。

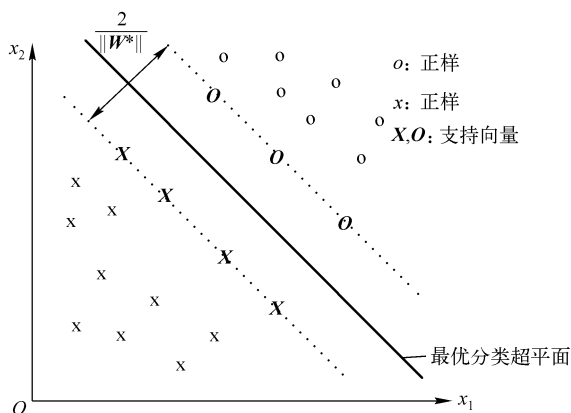


图 5-21 最优分类超平面

我们的目标是找到构造最优分类超平面的方法。考虑问题的一种等价表述: 找到一个向量 W^* 和常数 b^* , 使它们满足约束条件

$$\begin{cases} \langle X^+ \cdot W^* \rangle + b^* \geq 1 \\ \langle X^- \cdot W^* \rangle + b^* \leq -1 \end{cases} \quad (5-105)$$

且向量 W^* 具有最小范数

$$\min_p(W) = \frac{1}{2} \|W^*\|^2 \quad (5-106)$$

此时的判别函数为

$$\begin{aligned} f(X) &= W^* \cdot X + b^* \\ \begin{cases} \text{若 } f(X) > 0, & \text{则 } X \in X^+ \\ \text{若 } f(X) < 0, & \text{则 } X \in X^- \end{cases} \end{aligned} \quad (5-107)$$

定理 5.1 在满足式(5-105)的条件下最小化式(5-106)所得到的向量 W^* 与构成最优分类超平面的向量之间的关系是

$$\phi_0 = \frac{W^*}{\|W^*\|} \quad (5-108)$$

最优分类超平面与分类向量之间的间隔 γ_0 为

$$\gamma(\phi_0) = \sup \frac{1}{2} (c_1(\phi_0) - c_2(\phi_0)) = \frac{1}{\|\mathbf{W}^*\|} \quad (5-109)$$

因此,具有最小范数且满足约束式(5-105)的向量 \mathbf{W}^* 定义了最优分类超平面。

为了简化符号,将约束式(5-105)改写成如下形式:

$$y_i(\langle \mathbf{X}_i \cdot \mathbf{W}^* \rangle + b^*) \geq 1, i = 1, \dots, N \quad (5-110)$$

因此,为了得到最优分类超平面,我们必须求解二次规划问题,在线性约束式(5-110)条件下,最小化二次型,参见式(5-106)。经典的求解方法可用 Lagrange 乘子法求解, Lagrange 方程为

$$L(\mathbf{W}, \mathbf{a}, b) = \frac{1}{2} \|\mathbf{W}\|^2 - \sum_{i=1}^N a_i \{y_i(\langle \mathbf{X}_i \cdot \mathbf{W} \rangle + b) - 1\} \quad (5-111)$$

其中 $a_i \geq 0$ 为 Lagrange 乘子。对 \mathbf{W} 和 b 求偏微分,得到如下条件:

$$\begin{cases} \frac{\partial L(\mathbf{W}, \mathbf{a}, b)}{\partial \mathbf{W}} = \mathbf{W} - \sum_{i=1}^N y_i a_i \mathbf{X}_i = 0 \\ \frac{\partial L(\mathbf{W}, \mathbf{a}, b)}{\partial b} = - \sum_{i=1}^N y_i a_i = 0 \end{cases} \quad (5-112)$$

得到关系式

$$\mathbf{W} = \sum_{i=1}^N y_i a_i \mathbf{X}_i \quad (5-113)$$

$$0 = \sum_{i=1}^N y_i a_i \quad (5-114)$$

代入式(5-111)得到

$$\max \mathbf{H}(\mathbf{a}) = \sum_{i=1}^N a_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j a_i a_j \langle \mathbf{X}_i \cdot \mathbf{X}_j \rangle \quad (5-115)$$

因为上式只涉及向量 \mathbf{a} 的求解,因此我们把 $L(\mathbf{W}, \mathbf{a}, b)$ 改成 $\mathbf{H}(\mathbf{a})$, 此式也称为原问题的对偶问题。为了构造最优化超平面,在 $a_i \geq 0, i = 1, 2, \dots, N$ 且满足式(5-114)条件下,对式(5-115)求解得到 $a_i^* \geq 0, i = 1, 2, \dots, N$, 代入式(5-113)得到向量

$$\mathbf{W}^* = \sum_{i=1}^N y_i a_i^* \mathbf{X}_i \quad (5-116)$$

在求解此优化问题中, Karush- Kuhn- Tucker 互补条件提供了关于解结构的有用信息。根据 KKT 条件,最优解 \mathbf{a}^* 必须满足:

$$a^* [y_i(\langle \mathbf{W}^* \cdot \mathbf{X}_i \rangle + b^*) - 1] = 0, \quad i = 1, \dots, N \quad (5-117)$$

这意味着仅仅是函数间隔为 1 的样本向量 \mathbf{X}_i , 也就是最靠近超平面的点对应的 \mathbf{a}_i^* 非零, 称这样的向量为支持向量。在图 5-30 中大写字母所示样品向量为支持向量。所有其他样本向量对应的 \mathbf{a}_i^* 为零。因此在计算权重向量的表达式(5-116)中, 实际只有这些支持向量包括在内。

最优解 \mathbf{a}^* 和 \mathbf{W}^* 可由二次规划算法求得。选取一个支持向量 \mathbf{X}_i , 可求得 b^*

$$b^* = y_i - \langle \mathbf{X}_i \cdot \mathbf{W}^* \rangle \quad (5-118)$$

最优判别函数具有如下形式:

$$f(\mathbf{X}) = \sum_{i=1}^N y_i a_i^* \langle \mathbf{X}_i \cdot \mathbf{X} \rangle + b^* \quad (5-119)$$

(2) 支持向量机模型

我们介绍了线性可分的支持向量机最优超平面的求解, 而对于线性不可分的分类问题, 必

须对最优化问题做一些改动。可以通过非线性变换把样本输入空间转化为某个高维空间中的线性问题,在高维空间中求线性最优分类超平面,这样的高维空间也称为特征空间或高维特征空间(Hilbert 空间)。这种变换可能比较复杂,且高维特征空间的转换函数也很难显式地表示出来,因此这种思路在一般情况下不易实现。但是注意到,在线性情况下的对偶问题中,不论是寻优目标函数式(5-115)还是判别函数式(5-119)都只涉及训练样本之间的内积运算 $\langle \mathbf{X}_i \cdot \mathbf{X}_j \rangle$ 。

假设有非线性映射 $\Phi: R^n \rightarrow H$ 将输入空间的样本映射到高维(可能是无穷维)的特征空间 H 中。当在特征空间 H 中构造最优超平面时,训练算法仅使用空间中的点积,即 $\langle \Phi(\mathbf{X}_i), \Phi(\mathbf{X}_j) \rangle$, 而没有单独的 $\Phi(\mathbf{X}_i)$ 出现。因此,如果能够找到一个函数 k 使得 $k(\mathbf{X}_i, \mathbf{X}_j) = \langle \Phi(\mathbf{X}_i), \Phi(\mathbf{X}_j) \rangle$, 这样,在高维特征空间中实际上只需进行内积运算,而这种内积运算可以用输入空间中的某些特殊函数来实现,我们甚至没有必要知道变换 Φ 的具体形式。这些特殊的函数 k 称为核函数。根据泛函的有关理论,只要核函数 $k(x_i, x_j)$ 满足 Mercer 条件,它就对应某一变换空间中的内积。

此时,目标函数变为

$$\begin{aligned} \max \mathbf{H}(\mathbf{a}) &= \sum_{i=1}^N a_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j a_i a_j k(\mathbf{X}_i \cdot \mathbf{X}_j) \\ \text{subject to } &\sum_{i=1}^n y_i a_i = 0, a_i \geq 0, i = 1, \dots, N \end{aligned} \quad (5-120)$$

相应的判别函数变为

$$f(\mathbf{X}) = \sum_{i=1}^N y_i a_i^* k(\mathbf{X}_i \cdot \mathbf{X}) + b^* \quad (5-121)$$

这就是支持向量机(SVM)。

支持向量机利用输入空间的核函数取代了高维特征空间中的内积运算,解决了算法可能导致的“维数灾难”问题:在构造判别函数时,不是对输入空间的样本做非线性变换,然后在特征空间中求解;而是先在输入空间比较向量(如求内积或是某种距离),对结果再做非线性变换。这样,大的工作量将在输入空间而不是在高维特征空间中完成。

支持向量机判别函数形式上类似于一个神经网络,输出是 M 个中间结点的线性组合,每个中间结点对应一个支持向量,如图 5-22 所示。

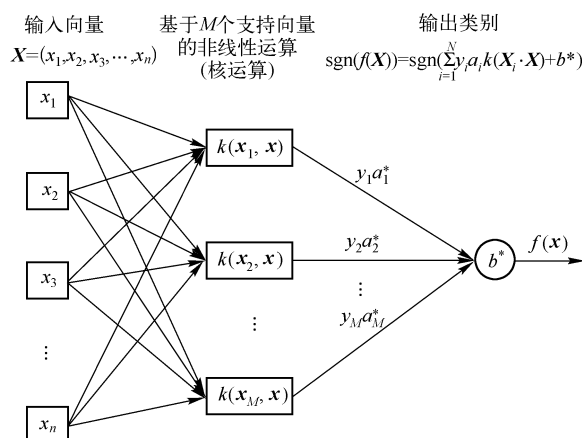


图 5-22 支持向量机的网络结构

(3) 特征空间与核函数

Mercer 定理将核解释为特征空间的内积,核函数的思想是将原本在高维特征空间中的计算,通过核函数在输入空间中完成,而无须知道高维变换的显式公式。为了避免维数灾难,许多学习算法都是通过“降维”的方式,将高维原始空间变换到较低维的特征空间,这容易损失一些有用的特征,导致学习性能的下降;而基于核的方法却恰好相反,它将低维向高维映射,却不需要过多地考虑维数对学习机器性能的影响。核函数是支持向量机的重要组成部分。根据 Hilbert-Schmidt 定理,只要变换 Φ 满足 Mercer 条件,就可用于构建核函数, Mercer 条件如下:给定对称函数 $k(x, \gamma)$ 和任意函数 $\varphi(x) \neq 0$, 满足约束

$$\begin{cases} \int_{-\infty}^{+\infty} \varphi^2(x) dx < 0 \\ \iint_{-\infty}^{+\infty} k(x, y) \varphi(x) \varphi(y) dx dy > 0 \end{cases} \quad (5-122)$$

目前常用的核函数主要有线性核函数(Linear)、二次核函数(Quadratic)、多项式核函数(Polynomial)、径向基核函数(Radial Basis Function, RBF)、多层感知器(Multi-Layer Perceptron, MLP),函数定义见式(5-67)~式(5-71)。

2. 实现步骤

要实现支持向量机的多类分类,首先要实现两类分类。支持向量机分类算法包括两部分,支持向量机的训练和支持向量机分类。

(1) 支持向量机训练的步骤

- ① 输入两类训练样品向量 $(\mathbf{X}_i, \mathbf{Y}_i) (i=1, 2, \dots, N, \mathbf{X} \in R^n, y \in \{-1, 1\})$, 类号分别为 ω_1, ω_2 。如果 $\mathbf{X}_i \in \omega_1$, 则 $y_i = -1$; $\mathbf{X}_i \in \omega_2$, 则 $y_i = 1$ 。
- ② 指定核函数类型。
- ③ 利用二次规划方法求解目标函数式(5-120)的最优解, 得到最优 Lagrange 乘子 a^* 。
- ④ 利用样本库中的一个支持向量 \mathbf{X} , 代入式(5-121), 左值 $f(\mathbf{X})$ 为其类别值 $(-1 \text{ 或 } 1)$, 可得到偏差值 b^* 。

(2) 支持向量机分类的步骤

- ① 输入待测样品 \mathbf{X} 。
- ② 利用训练好的 Lagrange 乘子 a^* 、偏差值 b^* 和核函数, 根据式 (5-119) 求解判别函数 $f(\mathbf{X})$ 。
- ③ 根据 $\text{sgn}(f(\mathbf{X}))$ 的值, 输出类别。如果 $\text{sgn}(f(\mathbf{X}))$ 为 -1 , 则该样品属于类 ω_1 ; 如果 $\text{sgn}(f(\mathbf{X}))$ 为 1 , 则该样品属于类 ω_2 。

3. 编程代码

Matlab 中的支持向量机工具箱中提供了支持向量机训练和支持向量机分类方法的函数, 其中 `svmtrain` 函数实现对样品进行训练并将训练结果保存到指定结构; `svmcclassify` 函数通过输入指定训练结果和待测样品实现待测样品的分类。具体函数说明可参考 Matlab 帮助文件。

(1) 支持向量机训练方法代码

%%%%%%%%%%%

```

% 函数名称:SupportVectorTrain( )
% 参数:
% 返回值:
% 函数功能:支持向量机训练
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function SupportVectorTrain( ) ;
    patternNum = 50 ;
    classnum = 0 ;
    % 选择核函数类型
    str = '线性核函数 ','二次核函数 ','多项式核函数 ','rbf 核函数 ','多层感知器核函数 ' ;
    kernalType = listdlg('ListString',str,'PromptString','选择核函数计算类型 ',
        'SelectionMode','Single','ListSize',[160,100],'Name','核函数选择对话框 ');
    switch( kernalType)
        case 1
            kernal = 'linear' ;
        case 2
            kernal = 'quadratic' ;
        case 3
            kernal = 'polynomial' ;
        case 4
            kernal = 'rbf' ;
        case 5
            kernal = 'mlp' ;
    end
    load templet pattern ;
    for i = 1 : 10
        for j = 1 : i - 1
            x = [ pattern(i).feature( :, 1 : patternNum) pattern(j).feature( :, 1 : patternNum) ] ;
            y = ones( 1 , patternNum * 2 ) ;
            y( patternNum + 1 : patternNum * 2 ) = - 1 ;
            % 进行两类支持向量机训练,结果保存到 svmStruct 结构中。
            svmStruct( i , j ) = svmtrain( x , y , 'Kernel _ Function' , kernal ) ;
        end
    end
    % 保存训练结果
    save svmStruct svmStruct ;
    msgbox('训练结束 ');

```

(2) 支持向量机分类方法代码

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 函数名称:SupportVector( )
% 参数:sample:待测样品
% 返回值:result:分类结果

```

```

% 函数功能:支持向量机分类
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function result = SupportVector( sample );
    % 读取训练结果
    load svmStruct svmStruct
    num = zeros( 1,10 );
    classnum = 0;
    for i = 1:10
        for j = 1:i - 1
            % 支持向量机两类分类
            G = svmclassify( svmStruct(i,j),sample );
            if( G == 1 )
                num(i) = num(i) + 1;
            elseif( G == -1 )
                num(j) = num(j) + 1;
            end
        end
    end
    % 找出分类数目最多的类
    [ max_val,max_pos ] = max( num );
    result = max_pos - 1;

```

4. 效果图

首先对样品库进行支持向量机训练,单击“支持向量机训练”菜单,弹出“核函数选择”对话框,如图 5-23(a) 所示。选择合适的核函数,开始训练,等待至训练结束(图 5-23(b))。手写一数字,如图 5-23(c) 所示,单击“支持向量机分类”菜单,进行分类,显示分类结果对话框,如图 5-23(d) 所示。



图 5-23 支持向量机分类效果图

本章小结

本章介绍了线性判别函数与非线性判别函数,并讨论了各种分类情况的判别函数设计方法、LMSE 验证可分性的方法,线性分类器介绍了感知器算法、增量校正算法、LMSE 分类算法,这三种算法的实现步骤大体相同,感知器算法只有在某一个模式样品被错误分类时才校正权

向量,而增量校正算法和 LMSE 算法的每一步都要校正权向量;同时还介绍了 Fisher 分类算法,它是将特征空间进行投影压缩后以实现分类的方法。非线性分类器介绍了核 Fisher 方法、势函数法和支持向量机方法。

习题 5

1. 写出每一个类别可用单个判别平面分开的判别函数形式。
2. 写出每两个类别之间可用判别平面分开的判别函数形式。
3. 简述多类可分的判别函数实现方法。
4. 写出非线性分类器判别函数的一般形式。
5. 简述感知器算法的分类准则,并写出奖惩算法的实现步骤。
6. 写出增量校正算法的实现步骤。
7. 简述增量校正算法与感知器算法的不同。
8. 简述 LMSE 算法验证可分性的基本原理。
9. 简述 Fisher 算法分类的实现步骤。
10. 简述基于核的 Fisher 算法分类的实现步骤。
11. 简述非线性分类器势函数算法实现分类的步骤。
12. 简述支持向量机的原理以及支持向量机分类方法的实现步骤。

第 6 章 神经网络分类器设计

本章要点:

- ☑ 人工神经网络的基本原理
- ☑ BP 神经网络
- ☑ 径向基函数神经网络(RBF)
- ☑ 自组织竞争神经网络
- ☑ 概率神经网络(PNN)
- ☑ 对向传播神经网络(CPN)
- ☑ 反馈型神经网络(Hopfield)

6.1 人工神经网络的基本原理

人工神经网络结构和工作机理基本上是以人脑的组织结构(大脑神经网络)和活动规律为背景的,它反映了人脑的某些基本特征,但并不是要对人脑部分的真实再现,可以说它是某种抽象、简化或模仿。参照生物神经网络发展起来的人工神经网络现已有许多种类型,但它们中的基本单元——神经元的结构是基本相同的。

6.1.1 人工神经元

人工神经元模型是生物神经元的模拟与抽象。这里所说的抽象是从数学角度而言,所谓模拟是以神经元的结构和功能而言的。如图 6-1 所示是一种典型的人工神经元模型,它是模拟生物神经元的细胞体、树突、轴突、突触等主要部分而构成的。

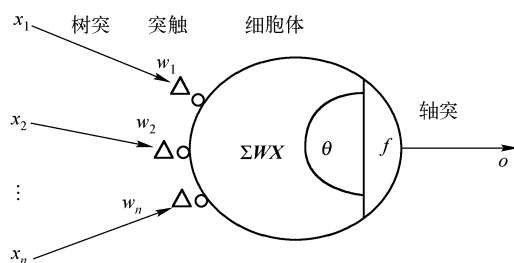


图 6-1 人工神经元模型

人工神经元相当于一个多输入单输出的非线性阈值器件。这里的 x_1, x_2, \dots, x_n 表示它的 n 个输入; w_1, w_2, \dots, w_n 表示与它相连的 n 个突触的连接强度,其值成为权值; ΣWX 称为激活值,表示这个人工神经元的输入总和,对应于生物神经细胞的膜电位; o 表示这个人工神经元的输出; θ 表示这个人工神经元的阈值。如果输入信号的加权和超过 θ ,则人工神经元被激活。

这样,人工神经元的输出可描述为

$$o = f(\sum \mathbf{W}\mathbf{X} - \theta) \quad (6-1)$$

式中, $f()$ 表示神经元输入/输出关系函数,称为激活函数或输出函数。

\mathbf{W} 为权矢量(weight vector):

$$\mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

\mathbf{X} 为输入矢量(input vector):

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

设 $net = \mathbf{W}^T \mathbf{X}$ 是权与输入的矢量积(标量),相当于生物神经元由外加刺激引起的膜内电位的变化。这样激活函数可写成 $f(net)$ 。

阈值 θ 一般不是一个常数,它是随着神经元的兴奋程度而变化的。

激活函数有许多种类型,其中比较常用的激活函数可归结为三种形式:阈值函数、Sigmoid函数和分段线性函数。

1. 阈值函数(Threshold Function)

阈值函数通常也称为阶跃函数。阈值函数定义为

$$f(t) = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0 \end{cases} \quad (6-2)$$

若激励函数采用阶跃函数,如图 6-2(a)所示的人工神经元模型即为著名的 MP(McCulloch-Pitts)模型。此时神经元的输出取 1 或 0,反映了神经元的兴奋或抑制。

此外,符号函数 $\text{sgn}(t)$ 也常常作为神经元的激励函数,如图 6-2(b)所示。

$$\text{sgn}(t) = \begin{cases} 1, & t \geq 0 \\ -1, & t < 0 \end{cases} \quad (6-3)$$

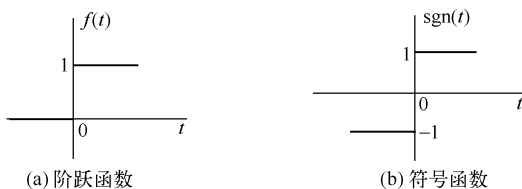


图 6-2 阈值函数

2. Sigmoid 函数(Sigmoid Function)

Sigmoid 函数也称为 S 型函数。到目前为止,它是人工神经网络中最常用的激励函数。S 型函数的定义为

$$f(t) = \frac{1}{1 + e^{-at}} \quad (6-4)$$

式中, a 为 Sigmoid 函数的斜率参数,通过改变参数 a ,我们会获取不同斜率的 Sigmoid 函数,如图 6-3 所示。

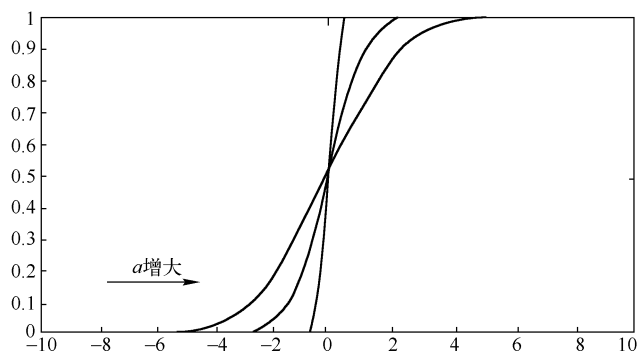


图 6-3 Sigmoid 函数

当斜率参数接近无穷大时,此函数转化为简单的阈值函数,但 Sigmoid 函数对应 0 ~ 1 的连续区域,而阈值函数对应的只是 0 和 1 两点,此外,Sigmoid 函数是可微分的,而阈值函数是不可微分的。

Sigmoid 函数也可用双曲正切函数(Signum Function)来表示:

$$f(t) = \tanh(t) \quad (6-5)$$

双曲正切函数如图 6-4 所示。

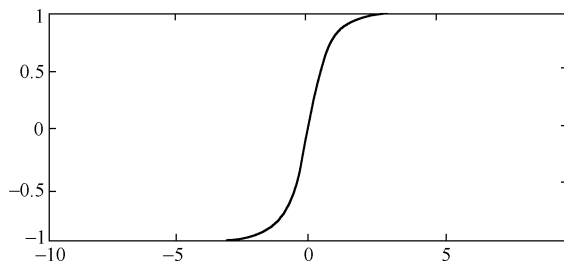


图 6-4 双曲正切函数

3. 分段线性函数(Piecewise-Linear Function)

分段线性函数定义为

$$f(t) = \begin{cases} 1, & t \geq 1 \\ t, & -1 < t < 1 \\ -1, & t \leq -1 \end{cases} \quad (6-6)$$

该函数在线性区间 $[-1, 1]$ 内的放大系数是一致的, 如图6-5所示。

这种形式的激励函数可看做是非线性放大器的近似。

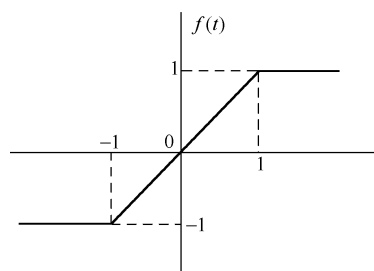


图6-5 分段线性函数

(1) 分段函数的特殊形式

以下两种情况是分段线性函数的特殊形式:

- ① 在执行中保持线性区域而使其不进入饱和状态, 则会产生线性组合器。
- ② 若线性区域的放大倍数无限大, 则分段线性函数简化为阈值函数。

(2) 神经元的特点

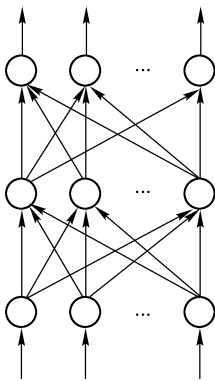
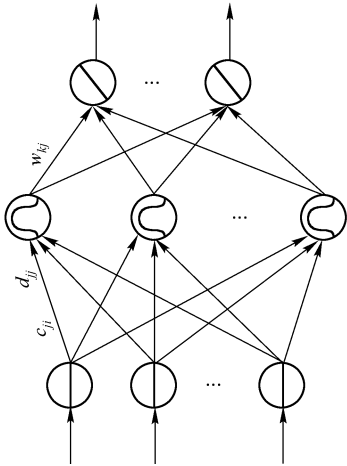
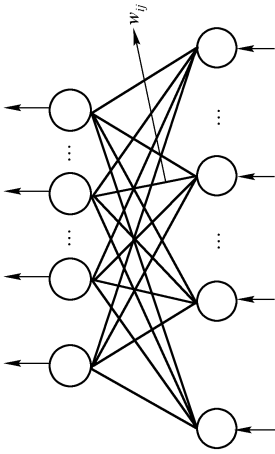
- ① 神经元是多输入、单输出元件。
- ② 它具有非线性的输入、输出特性。
- ③ 它具有可塑性, 可塑性反映在新突触的产生和现有神经突触的调整上, 可塑性使神经网络能够适应周围的环境。其塑性变化的部分主要是权值(w_i)的变化, 这相当于生物神经元的突触部分的变化。对于激发状态, w_i 取正值, 对于抑制状态, w_i 取负值。
- ④ 神经元的输出响应是各个输入值的综合作用结果。
- ⑤ 时空整合功能, 时间整合功能表现在不同时间、同一突触上; 空间整合功能表现在同一时间、不同突触上。
- ⑥ 兴奋与抑制状态, 当传入冲动的时空整合结果, 使细胞膜电位升高, 超过被称为动作电位的阈值, 细胞进入兴奋状态, 产生神经冲动, 由轴突输出; 同样, 当膜电位低于阈值时, 无神经冲动输出, 细胞进入抑制状态。

6.1.2 人工神经网络模型

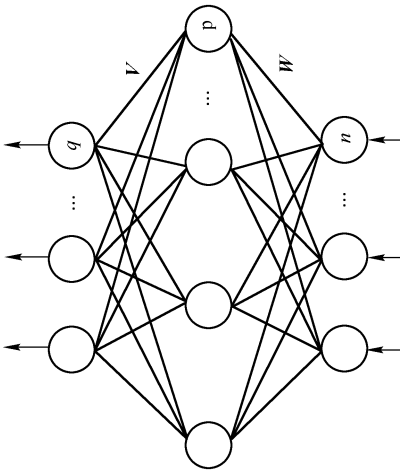
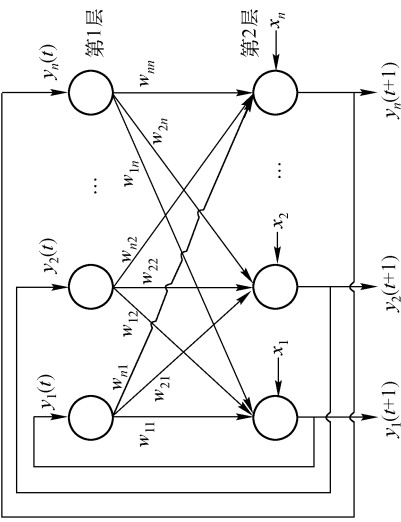
根据神经元之间连接的拓扑结构上的不同, 可将神经网络结构主要分为两大类, 即分层网络和相互连接型网络。分层网络是将一个神经网络中的所有神经元按功能分为若干层, 一般有输入层、隐含层和输出层, 各层顺序连接。分层网络可以细分为三种互连形式: 简单的前向网络、具有反馈的前向网络以及层内有相互连接的前向网络。对于简单的前向网络, 给定某一输入模式, 网络能产生一个相应的输出模式, 并保持不变。输入模式由输入层进入网络, 经过隐含层的模式变换, 由输出层产生输出模式。因此前向网络是由分层网络逐层模式变换处理的方向而得名的。相互连接型网络是指网络中任意两个单元之间都是可以相互连接的。对于给定的输入模式, 相互连接型网络由某一初始状态出发开始运行, 在一段时间内网络处于不断更新输出状态的变化过程中。如果网络设计得好, 最终可能会产生某一稳定的输出模式; 如果设计得不好, 网络也有可能进入周期性振荡或发散状态。

本章着重应用 BP 神经网络、径向基函数神经网络、自组织竞争神经网络、概率神经网络、对向传播神经网络和反馈型神经网络对手写数字进行分类识别。其中 5 种网络结构模型及分类特点如表 6-1 所示。

表 6-1 5 种神经网络模型比较

	三层 BP 神经网络	径向基函数神经网络	自组织竞争网络
网络模型图			
结构特点	BP 神经网络具有三层或三层以上的多层神经网络,上下各神经元之间无连接	与 BP 神经网络结构相似,但其隐含层神经元的核函数取为高斯核函数	由输入和竞争层构成的两层网络,没有隐含层,输入和竞争层之间的神经元实现双向连接,同时竞争层各个神经元之间还存在横向连接
训练学习方式比较	当一对样品提供给神经网络后,从输入层经过各中间层向输出层传播,在输出层获得响应。按照减少目标输出与实际误差的方向,采用负梯度下降等多种方法,从输出层经过中间层逐层修正连接权值,正确率不断提高	输入层到隐含层采用非线性映射,隐含层到输出层采用线性映射,具有最佳逼近,克服局部极小值的性能。神经元个数可能比 BP 神经网络多,训练时间比 BP 少	网络竞争层的各神经元通过竞争来获取输入模式的响应机会,最后仅有一个神经元成为竞争胜利者,并将与获胜神经元有关的各连接权值向着更有利于其竞争的方向发展
学习方式	两步都采用有导师学习	第一步为无导师学习,第二步为有导师学习	无导师自组织自学习
训练时间	长	较短	较短

续表

	对向传播神经网络	反馈型网络
网络模型图		
结构特点	由输入层、竞争层、输出层组成三层结构,输入层与竞争层构成 SOM 网络,竞争层与输出层构成基本竞争型网络	全连接型网络,属于单层反馈非线性网络,每一个结点的输出均反馈到其他结点的输入
训练学习方式比较	仅仅调整与竞争层获胜神经元相关的连接权重向量,既反映了输入模式的统计特性,又反映了输出模式的统计特性。输入、输出模式通过竞争层实现了相互映射,即网络具有双向记忆的性能	是一种循环神经网络,从输出到输入有反馈连接,这个反馈过程一直进行下去。如果网络能稳定收敛,则反馈与迭代的计算过程所产生的变化越来越小,一旦到达了稳定平衡状态,会输出稳定的恒值。具有联想记忆功能
学习方式	从整体上看属于有导师型的网络,而由输入层和竞争层构成的 SOM 网络属于无导师型网络	无导师学习
训练时间	较长	短

6.1.3 神经网络的学习过程

人的学习过程主要有三种:有导师学习、无导师学习和强化学习。模仿人的学习过程,人们提出了多种神经网络的学习方式,按学习方式进行神经网络模型分类,可以分为相应的三种,即有导师学习网络、无导师学习网络和强化学习网络。有导师型的学习或者说有监督型的学习是在有指导和考察的情况下进行的,如果学完了没有达到要求,那么就要再继续学习(重新学习)。无导师型的学习或者说无监督型的学习是靠学习者或者说神经系统本身自行完成的。学习是一个相对持久的变化过程,学习往往也是一个推理的过程,例如,通过经验也可以学习,学习是神经网络最重要的能力。

人工神经网络可从所需要的例子集合中学习、从输入与输出的映射中学习。对于有监督学习,是在已知输入模式和期望输出的情况下进行的学习。对应每一个输入,有导师提供的系统以实际响应与期望响应之间的差距作为测量误差,用来校正网络的参数(权值和阈值),输入—输出模式的集合称为这个学习模型的训练样品集合。

神经网络最大的特点就是它有学习的能力,在学习过程中,主要是网络连接权的值发生了相应的变化,学习到的内容也是记忆在连接权当中。

6.1.4 人工神经网络在模式识别问题上的优势

人工神经网络(Artificial Neural Networks, ANN),简称神经网络(NN),是对人脑或自然神经网络若干基本特性的抽象和模拟,是一种基于连接学说构造的智能仿生模型,是由大量神经元组成的非线性动力系统。

以生物神经网络为模拟基础的人工神经网络试图在模拟推理和自动学习等方面向前发展,使人工智能更接近人脑的自组织和并行处理功能,它在模式识别、聚类分析和专家系统等多方面显示出了新的前景和新的思路。神经网络可以看成是从输入空间到输出空间的一个非线性映射,它通过调整权重和阈值来“学习”或发现变量间的关系,实现对事务的分类。由于神经网络是一种对数据分布无任何要求的非线性技术,它能有效解决非正态分布、非线性的评价问题,因而受到广泛的应用。由于神经网络具有信息的分布存储,并行处理以及自学习能力等特点,所以它在信息处理、模式识别、智能控制等领域有着广泛的应用前景。近年来,神经网络已成为研究的热点,并取得了广泛的应用。

1. 神经网络的特点

(1) 固有的并行结构和并行处理

人工神经网络和人类的大脑类似,不但结构上是并行的,它的处理顺序也是并行的和同时的。在同一层内的处理单元都是同时操作的,即神经网络的计算功能分布在多个处理单元上。而一般计算机通常有一个处理单元,其处理顺序是串行的。

(2) 知识的分布存储

在神经网络中,知识不是存储在特定的存储单元中,而是分布在整个系统中,要存储多个知识就需要很多链接。在计算机中,只要给定一个地址就可得到一个或一组数据。在神经网络中要获得存储的知识则采用“联想”的办法,这类似人类和动物的联想记忆。人类根据联想

善于正确识别图形,人工神经网络也是这样。

(3) 容错性

人工神经网络具有很强的容错性。它可以从不完善的数据和图形中进行学习并做出决定。由于知识存在于整个系统中,而不只是在一个存储单元中,预定比例的结点不参与运算,对整个系统的性能不会产生重大的影响。能够处理那些有噪声或不完全的数据,具有泛化功能和很强的容错能力。

(4) 自适应性

自适应性根据所提供的数据,通过学习和训练,找出输入和输出之间的内在关系,从而求得问题的解,而不是依据对问题的经验知识和规则。因而具有自适应功能,这对于弱化权重确定人为因素是十分有益的。

(5) 模式识别能力

目前有各种各样的神经网络模型,其中有很多网络模型善于模式识别。模式识别是 ANN 最重要的特征之一。它不但能识别静态信息,对实时处理复杂的动态信息(随时间和空间变化的)也具有巨大潜力。模式识别往往是非常复杂的,各个因素之间相互影响,呈现出复杂的非线性关系,人工神经网络为处理这类非线性问题提供了强有力的工具。

2. 人工神经网络的优点

人工神经网络在模式识别问题上,相比其他传统方法的优势可以大致归结为以下三点:

- ① 要求对问题的了解较少。
- ② 可对特征空间进行较为复杂的划分。
- ③ 适用于高速并行处理系统来实现。

但是人工神经网络同其他理论一样也不是完美的,也有其固有的弱点,例如,需要更多的训练数据,在非并行处理系统中的模拟运行速度很慢,以及无法获取特征空间中的决策面等。

6.2 BP 神经网络

6.2.1 BP 神经网络的基本概念

1. BP 神经网络拓扑结构

BP 神经网络是一种具有三层或三层以上的多层神经网络,每一层都由若干个神经元组成,如图 6-6 所示,它的左、右各层之间各个神经元实现全连接,即左层的每一个神经元与右层的每个神经元都有连接,而上下各神经元之间无连接,如图 6-7 所示。BP 神经网络按有导师学习方式训练,当一对学习模式提供给网络后,其神经元的激活值将从输入层经各隐含层向输出层传播,在输出层的各神经元输出对应于输入模式的网络响应。然后,按减少希望输出与实际输出误差的原则,从输出层经各隐含层、最后回到输入层逐层修正各连接权。由于这种修正过程是从输出到输入逐层进行的,所以称它为“误差逆传播算法”。随着这种误差逆传播训练的不断进行,网络对输入模式响应的正确率也将不断提高。

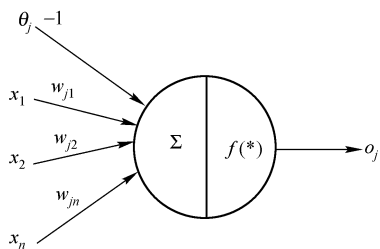


图 6-6 人工神经元模型

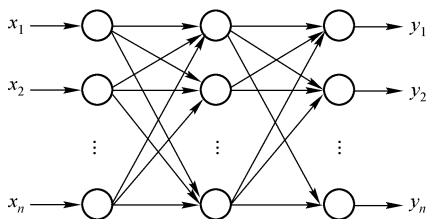


图 6-7 用于多指标综合评价的三层 BP 神经网络

由于 BP 神经网络有处于中间位置的隐含层,并有相应的学习规则可循,可训练这种网络,使其具有对非线性模式的识别能力。特别是它的数学意义明确、步骤分明的学习,更使其有广泛的应用前景。

2. BP 神经网络训练

在进行 BP 神经网络的设计时,应从网络的层数、每层中的神经元数、初始值以及学习速率等几个方面进行考虑。

① 网络的层数。已经证明:三层 BP 神经网络可以实现多维单位立方体 R^m 到 R^n 的映射,即能够逼近任何有理函数。这实际上给了一个设计 BP 神经网络的基本原则。增加层数可以更进一步地降低误差,提高精度,但同时也使网络复杂化,从而增加网络权值的训练时间。而误差精度的提高实际上也可以通过增加隐含层中的神经元数目来获得,其训练结果也比增加层数更容易观察和调整。所以一般情况下,应优先考虑增加隐含层中的神经元数。

② 隐含层的神经元数。网络训练精度的提高,可以通过采用一个隐含层而增加神经元数的方法来获得。这在结构的实现上要比增加更多的隐含层简单得多。在具体设计时,比较实际的做法是隐含层取输入层的两倍,然后适当地加上一余量。评价一个网络设计得好坏,首先是它的精度,其次是训练时间。时间包含有两层:一层是循环次数,二是每一次循环中计算所花的时间。

③ 初始权值的选取。由于系统是非线性的,初始值的选取对于学习是否达到局部最小、是否能够收敛以及训练时间的长短有很大关系。初始值过大、过小都会影响学习速度,因此权值的初始值应选为均匀分布的小数经验值,一般取初始权值在 $(-1, 1)$ 之间的随机数,也可选取在 $[-2.4/n, 2.4/n]$ 之间的随机数,其中 n 为输入特征个数。为避免每一步权值的调整方向是同向的,应将初始值设为随机数。

④ 学习速率。学习速率决定每一次循环训练中所产生的权值变化量。高的学习速率可能导致系统的不稳定;但低的学习速率导致较长的训练时间,可能收敛很慢,不过能保证网络的误差值跳出误差表面的低谷而最终趋于最小误差值。在一般情况下,倾向于选取较小的学习速率以保证系统的稳定性。学习速率的选取 $0.01 \sim 0.8$ 。

如同初始权值的选取过程一样,在一个神经网络的设计中,网络要经过几个不同的学习速率的训练,通过观察每一次训练后的误差平方和 $\sum e^2$ 的下降速率来判断所选定的学习速率是否合适,若 $\sum e^2$ 下降很快,则说明学习速率合适,若 $\sum e^2$ 出现震荡现象,这说明学习速率过大。对于每一个具体网络都存在一个合适的学习速率,但对于较复杂网络,在误差曲面的不同部位

可能需要不同的学习速率。为了减少寻找学习速率的训练次数以及训练时间,比较合适的方法是采用变化的自适应学习速率,使网络的训练在不同的阶段自动设置不同学习的速率。一般来说,学习速率越高,收敛越快,但容易震荡;而学习速率越低,收敛越慢。

⑤ 期望误差的选取。在网络的训练过程中,期望误差值也应当通过对比训练后确定一个合适的值。所谓的“合适”,是相对于所需要的隐含层的结点数来确定的,因为较小的期望误差要靠增加隐含层的结点,以及训练时间来获得。一般情况下,作为对比,可以同时两个不同期望误差的网络进行训练,最后通过综合因素的考虑来确定采用其中一个网络。

尽管含有隐含层的神经网络能实现任意连续函数的逼近,但在训练过程中如果一些参数选取得合适,可以加快神经网络的训练,缩短神经网络的训练时间和取得满意的训练结果。对训练过程有较大影响的是权系数的初值、学习速率等。

调整量与误差成正比,即误差越大,调整的幅度就越大,这一物理意义是显而易见的。

调整量与输入值的大小成正比例,这里由于输入值越大,在这次学习过程中就显得越活跃,所以与其相连的权值的调整幅度就应该越大。调整量与学习系数成正比。通常学习系数为 0.1~0.8,为使整个学习过程加快,又不引起振荡,可采用变学习速率的方法,即在学习初期取较大的学习系数,随着学习过程的进行逐渐减少其值。

3. BP 神经网络训练

为了使 BP 神经网络具有某种功能,完成某项任务,必须调整层间连接权值和结点阈值,使所有样品的实际输出和期望输出之间的误差稳定在一个较小的值以内。在训练 BP 神经网络的算法中,误差反向传播算法是最有效、最常用的一种方法。本章模式识别系统介绍了以下 9 种 BP 神经网络参数(连接权值和结点阈值)的调整方法:

- 梯度下降法;
- 有动量的梯度下降法;
- 有自适应 η 的梯度下降法;
- 有动量加自适应 η 的梯度下降法;
- 弹性梯度下降法;
- Fletcher-Reeves 共轭梯度法;
- Polak-Ribiere 共轭梯度法;
- Powell-Beale 共轭梯度法;
- 量化共轭梯度法。

三层 BP 神经网络学习训练过程如图 6-8 所示。

BP 神经网络的学习过程主要由四部分组成:

- ① 输入模式顺传播(输入模式由输入层经隐含层向输出层传播计算)。
- ② 输出误差逆传播(输出的误差由输出层经隐含层传向输入层)。
- ③ 循环记忆训练(模式顺传播与误差逆传播的计算过程反复交替循环进行)。
- ④ 学习结果判别(判定全局误差是否趋向极小值)。

下面以梯度下降法训练 BP 神经网络为例,介绍和分析这四个过程,在第 l 次输入样品 ($l=1,2,\dots,N$) 进行训练时各个参数的表达及计算方法。

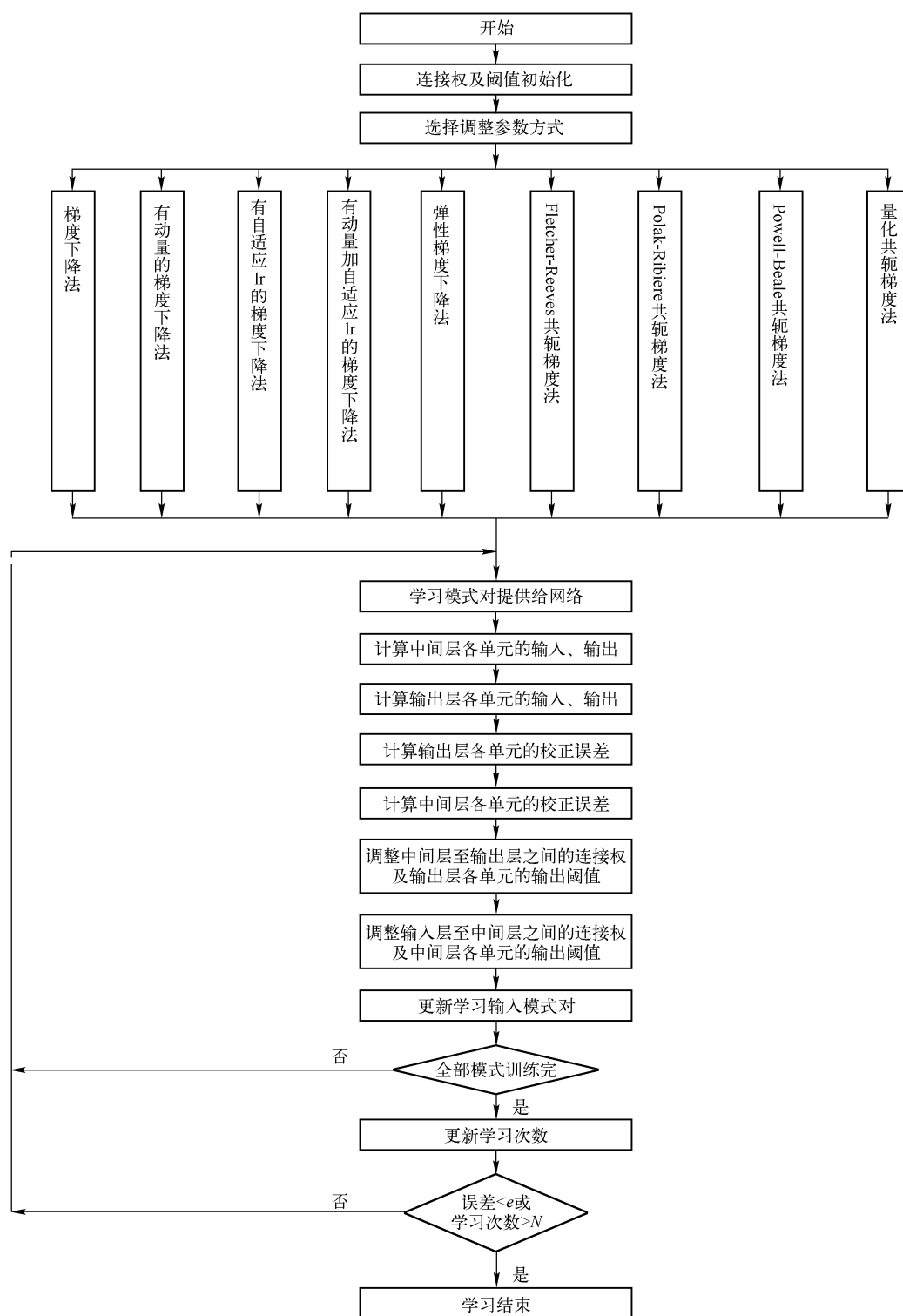


图 6-8 BP 神经网络学习训练过程

(1) 确定参数

① 确定输入向量 \mathbf{X} :

输入向量为 $\mathbf{X} = [x_1, x_2, \dots, x_n]^T$ (n ——输入层单元个数)。

② 确定输出向量 \mathbf{Y} 和希望输出向量 \mathbf{O} :

输出向量为 $\mathbf{Y} = [y_1, y_2, \dots, y_q]^T$ (q ——输出层单元数)。

希望输出向量 $\mathbf{O} = [o_1, o_2, \dots, o_q]^T$ 。

③ 确定隐含层输出向量 \mathbf{B} :

隐含层输出向量为 $\mathbf{B} = [b_1, b_2, \dots, b_p]^T$, (p ——隐含层单元数)。

④ 初始化输入层至隐含层的连接权值 $\mathbf{W}_j = [w_{j1}, w_{j2}, \dots, w_{ji}, \dots, w_{jn}]^T, j = 1, 2, \dots, p$ 。

⑤ 初始化隐含层至输出层的连接权值 $\mathbf{V}_k = [v_{k1}, v_{k2}, \dots, v_{kj}, \dots, v_{kp}]^T, k = 1, 2, \dots, q$ 。

(2) 输入模式顺传播

这一过程主要是利用输入模式求出它所对应的实际输出。

① 计算隐含层各神经元的激活值 s_j :

$$s_j = \sum_{i=1}^n w_{ji} \cdot x_i - \theta_j \quad (j = 1, 2, \dots, p) \quad (6-7)$$

式中, w_{ji} 为输入层至隐含层的连接权; θ_j 为隐含层单元的阈值。

激活函数采用 S 型函数, 即

$$f(x) = \frac{1}{1 + \exp(-x)} \quad (6-8)$$

这里之所以选 S 型函数为 BP 神经网络神经元的激活函数是因为它是连续可微分的, 而且更接近于生物神经元的信号输出形式。

② 计算隐含层 j 单元的输出值。将上面的激活值代入激活函数中可得隐含层 j 单元的输出值为

$$b_j = f(s_j) = \frac{1}{1 + \exp\left(-\sum_{i=1}^n w_{ji} \cdot x_i + \theta_j\right)} \quad (6-9)$$

阈值 θ_j 在学习过程中和权值一样也不断地被修正。阈值的作用反映在 S 函数的输出曲线上, 如图 6-9 所示。

由图中可见, 阈值的作用相当于将输出值移了 θ 个单位。

同理, 可求得输出端的激活值和输出值。

③ 计算输出层第 k 个单元的激活值 s_k :

$$s_k = \sum_{j=1}^p v_{kj} \cdot b_j - \theta_k \quad (6-10)$$

④ 计算输出层第 k 个单元的实际输出值 y_k :

$$y_k = f(s_k) \quad (t = 1, 2, \dots, q) \quad (6-11)$$

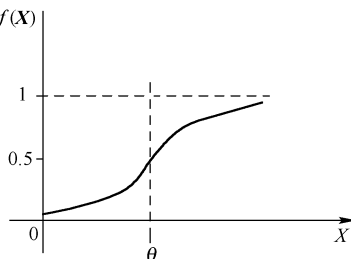


图 6-9 阈值的作用

式中, v_{kj} 为隐含层至输出层的权值; θ_k 为输出层单元阈值; $f(x)$ 为 S 型激活函数。

利用以上各式就可计算出一个输入模式的顺传播过程。

(3) 输出误差的逆传播

在第(2)步的模式顺传播计算中我们得到了网络的实际输出值, 当这些实际的输出值与希望的输出值不一样时或者说误差大于所限定的数值时, 就要对网络进行校正。

这里的校正从后向前进行的, 所以叫做误差逆传播, 计算时从输出层到隐含层, 再从隐含层到输入层。

① 输出层的校正误差为

$$d_k = (o_k - y_k) y_k (1 - y_k) \quad (k = 1, 2, \dots, q) \quad (6-12)$$

式中, y_k 为实际输出; o_k 为希望输出。

② 隐含层各单元的校正误差为

$$e_j = \left(\sum_{k=1}^q v_{kj} \cdot d_k \right) b_j (1 - b_j) \quad (6-13)$$

这里应注意, 每一个中间单元的校正误差都是由 q 个输出层单元校正误差传递而产生的。当校正误差求得后, 则可利用 d_k 和 e_j 沿逆方向逐层调整输出层至隐含层, 隐含层至输入层的权值。

③ 对于输出层至隐含层连接权和输出层阈值的校正量为

$$\Delta v_{kj} = \alpha \cdot d_k \cdot b_j \quad (6-14)$$

$$\Delta \theta_k = \alpha \cdot d_k \quad (6-15)$$

式中, b_j 为隐含层 j 单元的输出; d_k 为输出层的校正误差; α 为(学习系数), $\alpha > 0$ 。

④ 隐含层至输入层的校正量为

$$\Delta w_{ji} = \beta \cdot e_j \cdot x_i \quad (6-16)$$

$$\Delta \theta_j = \beta \cdot e_j \quad (6-17)$$

式中, e_j 为隐含层 j 单元的校正误差; β 为学习系数, $0 < \beta < 1$ 。

这里可以看出:

- 调整量与误差成正比, 即误差越大, 调整的幅度就越大, 这一物理意义是显而易见的。
- 调整量与输入值的大小成正比例, 这里由于输入值越大, 在这次学习过程中就显得越活跃, 所以与其相连的权值的调整幅度就应该越大。
- 调整量与学习系数成正比。通常学习系数为 0.1 ~ 0.8, 为使整个学习过程加快, 又不引起震荡, 可采用变学习速率的方法, 即在学习初期取较大的学习系数, 随着学习过程的进行逐渐减少其值。

(4) 循环记忆训练

为使网络的输出误差趋于极小值。对于 BP 神经网络输入的每一组训练模式, 一般要经过数百次甚至上万次的循环记忆训练, 才能使网络记住这一模式。这种循环记忆训练实际上

就是反复重复上面介绍的输入模式。

(5) 学习结果的判别

当每次循环记忆训练结束后,都要进行学习结果的判别。判别的目的是检查输出误差是否已经小到可以允许的程度。如果小到了可以允许的程度,就可以结束整个学习过程,否则还要进行循环训练。学习或者说训练的过程是网络全局误差趋向于极小值的过程。但是对于 BP 神经网络,其收敛过程存在着两个很大的缺陷:一是收敛速度慢,二是存在“局部极小点”问题。在学习过程中有时会出现,当学习反复进行到一定次数后,虽然网络的实际输出与希望输出还存在很大的误差,但无论再如何学习下去,网络全局误差的减少速度都变得很缓慢,或者根本不再变化,这种现象是因网络收敛于局部极小点所致。BP 神经网络的全局误差函数 E 是一个以 S 型函数为自变量的非线性函数。这就意味着由 E 构成的连接权空间不是只有一个极小点的曲面,而是存在多个局部极小点的超曲面,如图 6-10 所示。

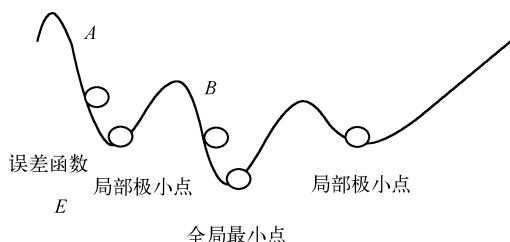


图 6-10 最小点和极小点

导致这一缺陷的主要原因是采用了按误差函数梯度下降的方向进行校正。在图 6-10 中,若初始条件是从 A 点的位置开始则只能达到局部极小点,但如果从 B 点开始则可达到全局最小点。所以 BP 神经网络的收敛依赖于学习模式的初始化位置,适当改进 BP 神经网络隐含层的单元个数,或者给每个连接权加上一个很小的随机数,都有可能使收敛过程避开局部极小点。

6.2.2 BP 神经网络分类器设计

1. BP 神经网络分类器结构设计

我们设计的 BP 神经网络结构有三层:输入层、隐含层、输出层,如图 6-11 所示。对于手写数字提取了 $5 \times 5 = 25$ 个特征,作为神经网络的输入,因此输入结点为 25 个,根据隐含层个数大约为输入结点两倍关系,隐含层取 50 个结点,输出层取 4 个结点,这 4 个输出为四位二进制数,代表神经网络输出的数字类型。

三层 BP 神经网络结构图如图 6-11 所示。图中所示的三层 BP 神经网络的学习分为正向传播输出和反向传播修正权值两阶段。

2. BP 神经网络调整参数 Matlab 实现方法

为了介绍 BP 神经网络分类识别方法,以手写数字模式识别为例,在 BP 神经网络结构基

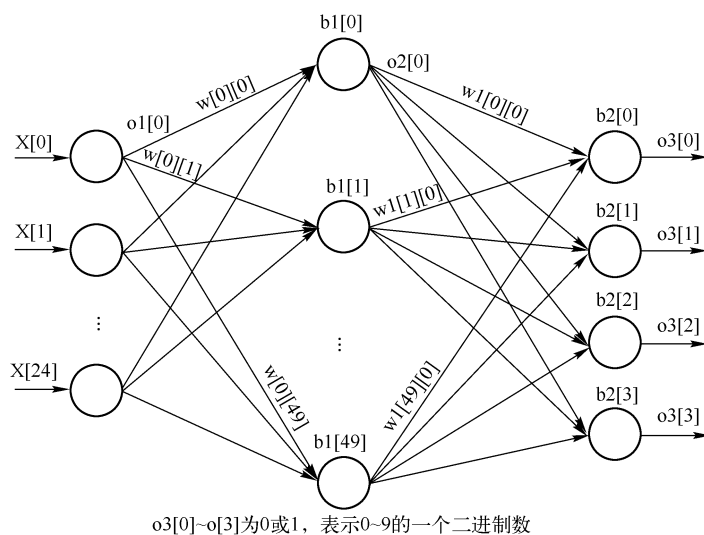


图 6-11 三层 BP 神经网络结构图

基础上,分别采用9种方式调整参数(连接权值和结点阈值),如表6-2所示。

(1) 梯度下降法

在样品被训练之前,需要构建 BP 网络,Matlab 中 newff 函数具有构建 BP 网络功能,该函数最后一个参数代表调整 BP 神经网络连接权值和阈值的方法。将最后一个参数设置为“traingd”,代表用梯度下降法调整 BP 参数,与“traingd”有关的调整参数如表 6-3 所示。

表 6-2 BP 神经网络常用调整参数方式

调整参数方式	标志码
梯度下降法	traingd
有动量的梯度下降法	traingdm
有自适应 lr 的梯度下降法	traingda
有动量加自适应 lr 的梯度下降法	traingdx
弹性梯度下降法	trainrp
Fletcher-Reeves 共轭梯度法	traincgf
Polak-Ribiere 共轭梯度法	traincgp
Powell-Beale 共轭梯度法	traincgb
量化共轭梯度法	trainscg

表 6-3 梯度下降法调整参数

参 数	描 述
net.trainParam. epochs	最大训练次数(默认为 10)
net.trainParam. goal	训练要求精度(默认为 0)
net.trainParam. lr	学习速率(默认为 0.01)
net.trainParam. max _ fail	最大失败次数(默认为 5)
net.trainParam. min _ grad	最小梯度要求(默认为 1e-10)
net.trainParam. show	显示训练迭代过程(NaN 表示不显示,默认为 25)
net.trainParam. time	最大训练时间(默认为 inf)

其中,学习速率是重要的参数,它和负梯度的乘积决定了权值和阈值的调整量,学习速率越大,调整步伐越大。学习速率过大,算法会变得不稳定;但是如果学习速率过小,算法收敛的时间就会增大。训练过程中,只要满足下面四个条件之一,训练就会停止:

- ① 超过最大迭代次数 epochs。
- ② 表现函数值小于误差指标 goal。

- ③ 梯度值小于要求精度 min_grad。
- ④ 训练所用时间超过时间限制 time。
- ⑤ 最大失败次数超过次数限制 max_fail。

(2) 有动量的梯度下降法

动量法降低了网络对于误差曲面局部细节的敏感性;梯度下降法在修正权值时,只是按照 k 时刻的负梯度方向修正,并没有考虑到以前积累的经验,即以前时刻的梯度方向,从而常常使学习过程发生震荡,收敛缓慢。为此,有人提出了如下的改进算法:

$$\omega_{ji}(t+1) = \omega_{ji}(t) + \eta[(1-\alpha)d(t) + \alpha d(t-1)] \tag{6-18}$$

式中, $d(t)$ 表示 t 时刻的负梯度; $d(t-1)$ 表示 $t-1$ 时刻的负梯度; η 为学习速率; $\alpha \in [0,1]$ 是动量因子,当 $\alpha=0$ 时,权值修正只与当前负梯度有关,当 $\alpha=1$ 时,权值修正就完全取决于上一次循环的负梯度了。

这种方法所加入的动量项实质上相当于阻尼项,它减小了学习过程的震荡,从而改善了收敛性。将 Matlab 中 newff 函数最后一个参数设为“traingdm”,代表用有动量梯度下降法调整参数。与“traingdm”有关的调整参数如表 6-4 所示。

表 6-4 有动量的梯度下降法参数

参 数	描 述
net.trainParam.epochs	最大训练次数(默认为 10)
net.trainParam.goal	训练要求精度(默认为 0)
net.trainParam.lr	学习速率(默认为 0.01)
net.trainParam.max_fail	最大失败次数(默认为 5)
net.trainParam.min_grad	最小梯度要求(默认为 1e-10)
net.trainParam.show	显示训练迭代过程(NaN 表示不显示,默认为 25)
net.trainParam.time	最大训练时间(默认为 inf)
net.trainParam.mc	动量因子(默认为 0.9)

(3) 有自适应 lr 的梯度下降法

学习速率对于整个训练过程有很大的影响,训练成功与否与学习速率的选取关系很大。如果在训练过程中合理地改变学习速率,会避免以上的缺点。有自适应 lr 的梯度下降法就能够自适应调整学习速率,从而增加稳定性,提高速度和精度。将 Matlab 中 newff 函数最后一个参数设为“traingda”,代表用有自适应 lr 的梯度下降法调整参数,与“traingda”有关的调整参数如表 6-5 所示。

表 6-5 有自适应 lr 的梯度下降法参数

参 数	描 述
net.trainParam.epochs	最大训练次数(默认为 10)
net.trainParam.goal	训练要求精度(默认为 0)
net.trainParam.lr	学习速率(默认为 0.01)

续表

参 数	描 述
net.trainParam.max_fail	最大失败次数(默认为5)
net.trainParam.min_grad	最小梯度要求(默认为1e-10)
net.trainParam.show	显示训练迭代过程(NaN表示不显示,默认为25)
net.trainParam.time	最大训练时间(默认为inf)
net.trainParam.lr_inc	学习速率lr增长比(默认为1.05)
net.trainParam.lr_dec	学习速率下降比(默认为0.7)
net.trainParam.max_perf_inc	表现函数增加最大化(默认为1.04)

(4) 有动量加自适应lr的梯度下降法

在有动量的梯度下降法中赋予自适应lr,得到有动量加自适应lr的梯度下降法。将Matlab中newff函数最后一个参数设为“traingdx”,代表用有动量加自适应lr的梯度下降法调整参数。与“traingdx”有关的调整参数如表6-6所示。

表 6-6 有动量加自适应lr的梯度下降法参数

参 数	描 述
net.trainParam.epochs	最大训练次数(默认为10)
net.trainParam.goal	训练要求精度(默认为0)
net.trainParam.lr	学习速率(默认为0.01)
net.trainParam.max_fail	最大失败次数(默认为5)
net.trainParam.min_grad	最小梯度要求(默认为1e-10)
net.trainParam.show	显示训练迭代过程(NaN表示不显示,默认为25)
net.trainParam.time	最大训练时间(默认为inf)
net.trainParam.lr_inc	学习速率lr增长比(默认为1.05)
net.trainParam.lr_dec	学习速率下降比(默认为0.7)
net.trainParam.max_perf_inc	表现函数增加最大化(默认为1.04)
net.trainParam.mc	动量因子(默认为0.9)

(5) 弹性梯度下降法

多层神经网络的隐含层大多采用Sigmoid型传递函数,这类函数又称为“挤压”函数。因为它们将没有边界限制的输入信号压缩到有限的输出范围之内,当输入量很大或者很小时,输出函数的斜率接近0。那么当应用梯度下降法训练多层网络时,其梯度数量级会很小,从而使得权值和阈值的调整范围减小,也就是说即使没有达到最优值,也会形成训练停止的结果。弹性梯度下降法就能够消除这种影响。

应用弹性梯度下降法训练BP神经网络时,权值修正取决于表现函数导数的正负号,而导数的数量级对权值修正并没有影响。其原理如下:权值变化的大小由不同的修正值决定。当前两次训练时导数的正负号没有改变,权值和阈值的修正值随着参数delt_inc增加;当前两次训练时导数的正负号有改变,权值和阈值的修正值随着参数delt_dec减小;如果导数为0,

则修正值不变。这样调整的结果是,权值的变化呈现震荡变化的数量级。将 Matlab 中 newff 函数最后一个参数设为“trainrp”,代表用弹性梯度下降法调整参数。与“trainrp”有关的调整参数如表 6-7 所示。

表 6-7 弹性梯度下降法参数

参 数	描 述
net.trainParam.epochs	最大训练次数(默认为 100)
net.trainParam.goal	训练要求精度(默认为 0)
net.trainParam.lr	学习速率(默认为 0.01)
net.trainParam.max_fail	最大失败次数(默认为 5)
net.trainParam.min_grad	最小梯度要求(默认为 1e-6)
net.trainParam.show	显示训练迭代过程(NaN 表示不显示,默认为 25)
net.trainParam.time	最大训练时间(默认为 inf)
net.trainParam.delt_inc	权值变化增加量(默认为 1.2)
net.trainParam.delt_dec	权值变化减小量(默认为 0.5)
net.trainParam.delt0	初始权值变化(默认为 0.07)
net.trainParam.deltmax	权值变化最大值(默认为 50)

(6) 共轭梯度法

共轭梯度法是梯度法的一种改进方法,可以改进梯度法震荡和收敛性差的缺点。其基本思想是寻找与负梯度方向和上一次搜索方向共轭的方向作为新的搜索方向,从而加快训练速度,并提高训练精度。

所有的共轭梯度方法,都采用负梯度方向作为初始搜索方向:

$$p(0) = -g(0) \quad (6-19)$$

然后,沿着该方向作一维搜索:

$$w(t+1) = w(t) + \alpha(t)p(t) \quad (6-20)$$

接下来,就利用共轭方向作为新一轮的搜索方向,通常在当前负梯度方向上附加上一次搜索方向:

$$p(t) = -g(t) + \beta(t)p(t-1) \quad (6-21)$$

β 的选取衍生出各种共轭梯度法,如 Fletcher-Reeves、Polak-Ribiere、Powell-Beale 等修正方法。

共轭梯度法通常比自适应 lr 的梯度下降法速度快,有时候也优于弹性梯度下降法。同时,由于共轭梯度法占用较少的存储空间,因此在训练复杂网络的时候,通常选用共轭梯度法。

① Fletcher-Reeves 共轭梯度法。共轭梯度法中 Fletcher-Reeves 修正系数如下:

$$\beta(t) = \frac{g^T(t)g(t)}{g^T(t)g(t-1)} \quad (6-22)$$

将 Matlab 中 newff 函数最后一个参数设为“traincgf”,代表用 Fletcher-Reeves 共轭梯度法调整参数。与“traincgf”有关的调整参数如表 6-8 所示。与搜索函数有关的参数有 scal_tol、alpha、beta、delta、gama、low_lim、up_lim、maxstep、minstep、bmax,此处不再一一介绍。

表 6-8 Fletcher-Reeves 共轭梯度法参数

参 数	描 述
net.trainParam.epochs	最大训练次数(默认为 100)
net.trainParam.goal	训练要求精度(默认为 0)
net.trainParam.max_fail	最大失败次数(默认为 5)
net.trainParam.min_grad	最小梯度要求(默认为 1e-6)
net.trainParam.show	显示训练迭代过程(NaN 表示不显示,默认为 25)
net.trainParam.time	最大训练时间(默认为 inf)
net.trainParam.searchFcn	权值变化最大值(默认为 srchcha)

② Polak- Ribiere 共轭梯度法。共轭梯度法中 Polak- Ribiere 修正系数如下：

$$\beta(t)=\frac{\Delta g^T(t-1)g(t)}{g^T(t-1)g(t-1)}$$

(6-23)

将 Mablal 中 newff 函数最后一个参数设为“traincgp”,代表用 Polak- Ribieve 共轭梯度法调整参数。与“traincgp”有关的调整参数同 Fletcher- Reeves 共轭梯度法。

③ Powell- Beale 共轭梯度法。Powell- Beale 共轭梯度法和以上两种不同,因为它不是改变修正系数,而是改变设置负梯度方向的条件。将 Mablal 中 newff 函数最后一个参数设为“traincgb”,代表用 Powell- Beale 共轭梯度法调整参数。与“traincgb”有关的调整参数同 Fletcher- Reeves 共轭梯度法。

(7) 量化共轭梯度法

前面介绍的共轭梯度法每一步都需要进行一维搜索,这样就会耗费很多时间。应用 Moller 提出的量化共轭梯度法(SCG)融合了可信区间法和共轭梯度法,避免了耗时的一维搜索。

将 Mablal 中 newff 函数最后一个参数设为“trainscg”,代表用量化共轭梯度法调整参数。与“trainscg”有关的调整参数如表 6-9 所示。

表 6-9 量化共轭梯度法参数

参 数	描 述
net.trainParam.epochs	最大训练次数(默认为 100)
net.trainParam.goal	训练要求精度(默认为 0)
net.trainParam.max_fail	最大失败次数(默认为 5)
net.trainParam.min_grad	最小梯度要求(默认为 1e-6)
net.trainParam.show	显示训练迭代过程(NaN 表示不显示,默认为 25)
net.trainParam.time	最大训练时间(默认为 inf)
net.trainParam.sigma	因为二次求导对权值调整的影响参数(默认为 5.0e-5)
net.trainParam.lambda	Hessian 阵不确定性调节参数(默认为 5.0e-7)

3. 实现步骤

① 初始化输入、输出矩阵 $p[],t[]$ 。 p 为训练样品, t 为训练样品所属的类别。

② 构建 BP 神经网络,设置参数调整方式。Mablab 的 newff 函数具有构建 BP 神经网络的功能,为了选择不同的调整 BP 网络参数方式,只需修改 newff 函数最后一个参数,如表 6-10 所示,列出了采用不同的调整参数方式构建 BP 网络,newff 中的第一个参数代表手写数字训练集中输入结点为 25 个特征范围,第二个参数[50,4]代表隐含层和输出层的结点个数,由于构建 BP 网络和训练网络编程语句相同,本程序仅列出梯度下降法构建 BP 神经网络编程代码,供读者参考,其他不一一列出。

表 6-10 构建 BP 神经网络的核心函数

序号	调整 BP 网络参数方式	核 心 语 句
1	梯度下降法	bpnet = newff(x,[50,4],{ 'logsig','logsig' }, 'traingd');
2	有动量的梯度下降法	bpnet = newff(x,[50,4],{ 'logsig','logsig' }, 'traingdm');
3	有自适应 lr 的梯度下降法	bpnet = newff(x,[50,4],{ 'logsig','logsig' }, 'traingda');
4	有动量加自适应 lr 的梯度下降法	bpnet = newff(x,[50,4],{ 'logsig','logsig' }, 'traingdx');
5	弹性梯度下降法	bpnet = newff(x,[50,4],{ 'logsig','logsig' }, 'trainrp');
6	Fletcher-Reeves 共轭梯度法	bpnet = newff(x,[50,4],{ 'logsig','logsig' }, 'traincgf');
7	Polak- Ribiere 共轭梯度法	bpnet = newff(x,[50,4],{ 'logsig','logsig' }, 'traincgp');
8	Powell- Beale 共轭梯度法	bpnet = newff(x,[50,4],{ 'logsig','logsig' }, 'traincgb');
9	量化共轭梯度法	bpnet = newff(x,[50,4],{ 'logsig','logsig' }, 'trainscg');

③ 调用 Matlab 的 train(*bpnet*,*p*,*t*)函数,训练 BP 神经网络。其中,*bpnet* 为已经建立好的 BP 网络,*p* 为训练样品,*t* 为训练样品所属的类别。

④ 对待测样品,调用 Matlab 的 *sim* 函数,利用已经训练好的 *BP* 神经网络识别。*sim* 函数定义为:[*t*,*x*,*y*] = *sim*(*model*,*timespan*,*options*,*ut*);其中参数 *model* 表示网络结构名,*timespan* 表示循环次数,*options* 表示可选条件,*ut* 表示输入的向量,*t* 表示网络输出向量结构,*x* 表示仿真状态矩阵,*y* 表示仿真输出矩阵。

4. 编程代码

```
%%%%%%%%%%
% 函数名称:bpgdtrain
% 函数功能:构建 BP 神经网络,使用梯度下降法训练 BP 神经网络
% 函数参数:无
% 函数返回值:无
%%%%%%%%%
function bpgdtrain
    global bpnet;
    clc;
    load templet pattern;
    c = 0;
```

```

p = [];
for i = 1:10
    for j = 1:20
        c = c + 1;
        p(:,c) = pattern(i).feature(:,j);
    end
end
t = zeros(4,200);
t(4,1:20) = 1;
t(3,21:40) = 1;
t(3,41:60) = 1;
t(2,61:80) = 1;
t(2,81:100) = 1;
t(4,81:100) = 1;
t(2,3,101:120) = 1;
t(2,4,121:140) = 1;
t(1,141:160) = 1;
t(1,161:180) = 1;
t(4,161:180) = 1;
t(1,181:200) = 1;
t(3,181:200) = 1;
x = ones(25,2);
x(:,1) = 0;
bpnet = newff(x,[50,4],{'logsig','logsig'},'traingd');
bpnet.trainParam.show = 50; % 显示训练迭代过程(每隔 50 次训练,显示一次训练进程)
bpnet.trainParam.lr = 0.2; % 学习速率
bpnet.trainParam.epochs = 20000; % 最大训练次数
bpnet.trainParam.goal = 0.5e-1; % 训练要求的精度(科学计数法:0.05)
[bpnet] = train(bpnet,p,t);
%%%%
% 函数名称:bpnet
% 函数功能:识别手写数字
% 函数参数:手写数字特征:sample
% 函数返回值:手写数字所属类别:y
%%%%
function y = bpnet(sample)
    global bpnet;
    clc;
    a = sim(bpnet,sample');
    a = round(a);
    b = num2str(a);
    c = bin2dec(b');
    y = c - 1;

```

5. 效果图

① 选择“神经网络”→“BP 神经网络分类法”→“梯度下降法 BP 训练”菜单命令(注:此处也可选择使用其他方法,如有动量的梯度下降法、Powell-Beale 共轭梯度法等,训练 BP 神经网络),建立并训练 BP 神经网络如图 6-12 所示,训练结果如图 6-13 所示。



图 6-12 BP 神经网络梯度下降法训练

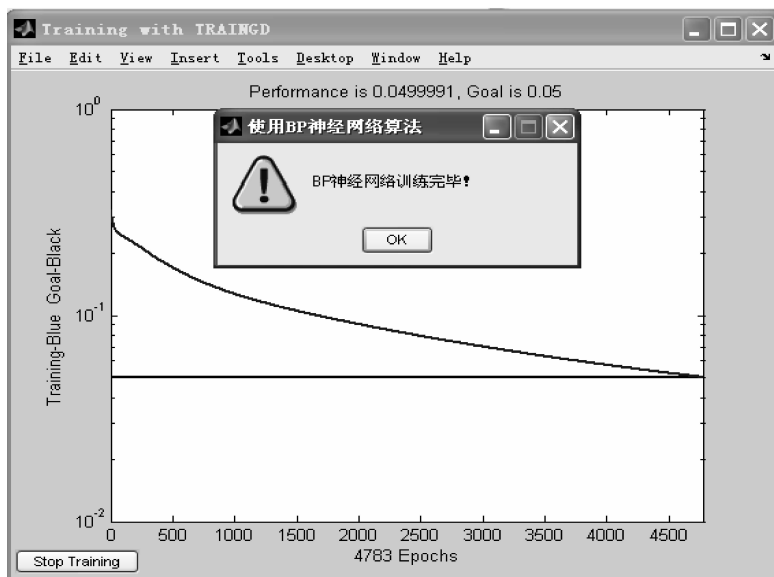


图 6-13 BP 神经网络梯度下降法训练结果

② 训练后权值详见文件《BP 神经网络训练后的权值和阈值.txt》。

③ 拖动鼠标左键在视图区用鼠标手写一个数字,如图 6-14 所示,然后选择“神经网络”→“BP 神经网络分类法”→“BP 网络法识别”菜单命令,进行手写数字分类。

由于样品库中存储的是作者自己手写的数字,神经网络训练集能够对这些形状的数字识别,如图 6-15 所示。读者可以借鉴下面各数字的形状来运行神经网络分类,会得到正确的结

果,否则读者书写的数字有可能分错。读者可以将自己手写的数字尽可能多的添加到样品库,每个数字添加 20 个左右即可容易识别读者个人手写的数字。

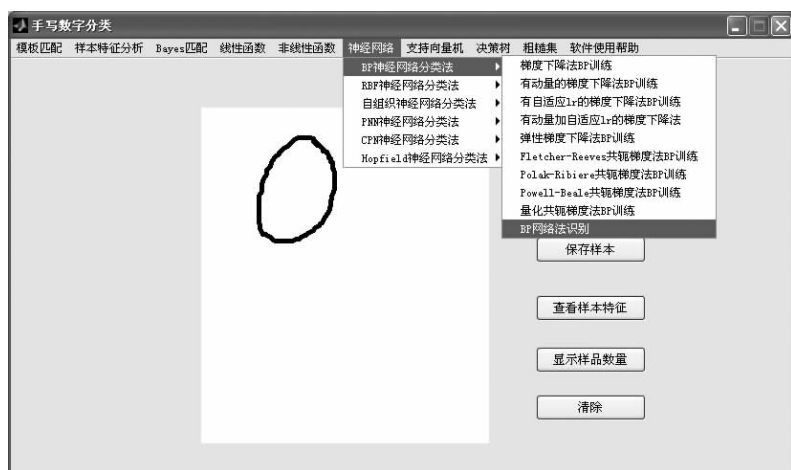


图 6-14 BP 神经网络识别菜单



(a) 手写数字0识别示意图



(b) 手写数字1识别示意图



(c) 手写数字2识别示意图



(d) 手写数字3识别示意图



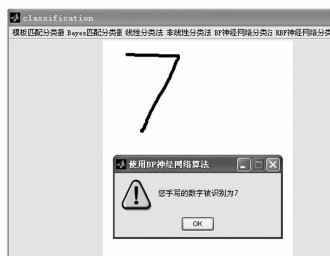
(e) 手写数字4识别示意图



(f) 手写数字5识别示意图



(g) 手写数字6识别示意图



(h) 手写数字7识别示意图



(i) 手写数字8识别示意图

图 6-15 BP 神经网络识别效果图



(j) 手写数字9识别示意图

图 6-15 BP 神经网络识别效果图(续)

6.3 径向基函数神经网络(RBF)

众所周知,BP 神经网络用于函数逼近时,权值的调节采用的是负梯度下降法。这种调节权值的方法具有局限性。本节主要介绍逼近能力、分类能力和学习速度等方面都优于 BP 神经网络的另一种网络——径向基函数神经网络(Radial Basis Function,RBF)。

径向基函数 RBF 神经网络(简称径向基网络)是由 J. Moody 和 C. Darken 于 20 世纪 80 年代末提出的一种神经网络结构,它是具有单隐层的三层前向网络。目前已经证明,径向基网络能够以任意精度逼近任意连续函数。

RBF 神经网络是一种性能良好的前向网络,具有最佳逼近,及克服局部极小值问题的性能。另外,基于 BP 网络的初始权值参数是随机产生的,而 RBF 网络的有关参数(如:具有重要性能的隐含层神经元的中心向量和宽度向量)则是根据训练集中的样本模式按照一定的规则来确定或者初始化的。这就可能使 RBF 神经网络在训练过程中不易陷入局部极小值的解域中。如果要实现同一个功能,径向基神经网络的神经元个数可能要比前向 BP 神经网络的神经元个数要多,但是,径向基网络所需要的训练时间却比 BP 神经网络少。

6.3.1 径向基函数神经网络的基本概念

用 RBF 作为隐单元的“基”构成隐含层空间,这样就可以将输入矢量直接(即不通过权连接)映射到隐空间。当 RBF 的中心点确定以后,这种映射关系也就确定了。而隐含层空间到输出空间的映射是线性的,即网络的输出是隐单元输出的线性加权和,此处的权即为网络可调参数。以上便是构成 RBF 网络的基本思想。由此可见,从总体上看,网络由输入到输出的映射是非线性的,而网络输出对可调参数而言却又是线性的;这样网络的权就可由线性方程组直接解出或用 RLS 方法递推计算,从而大大加快学习速度并避免局部极小问题。下面对这种网络进行介绍。

1. RBF 神经网络中心选取方法

对于 RBF 神经网络的学习算法,关键问题是隐含层神经元中心参数的合理确定。在已有的常用学习算法中,中心参数(或者中心参数的初始值)要么是从给定的训练样本集里按照某种方法直接选取,要么采用聚类的方法进行确定。RBF 神经网络中心选取常用方法有如下几种。

(1) 直接计算法(随机选取 RBF 中心)

这是一种最简单的方法。在此方法中,隐含层神经元的中心是随机地在输入样本中选取,且中心固定。一旦中心固定下来后,隐含层神经元的输出便是已知的,这样神经网络的连接权值就可以通过求线性方程组来确定。当样本数据的分布具有明显的代表性时,这种方法是一种简单有效的方法。

(2) 自组织学习选取 RBF 中心法

在这种方法中,RBF 神经网络的中心是可以变化的,并通过自组织学习确定其位置。而输出层的线性权重则是通过有监督的学习来确定的。因此,这是一种回合的学习方法。该方法在某种意义上是对神经网络资源的再分配,通过学习,使 RBF 的隐含层神经元中心位于输入空间重要的区域。这种方法主要采用 K-均值聚类法来选择 RBF 的中心,属于无监督(导师)的学习方法,在模式识别中有较为广泛的应用。

(3) 有导师学习选取 RBF 中心法

RBF 神经网络的中心以及其他参数都是通过有导师学习来确定的。通过训练样本集来获得满足导师(监督)要求的网络中心和其他权重参数,这也是 RBF 神经网络最一般的学习方法。常用的学习迭代方法是梯度下降法。

(4) 正交最小二乘法选取 RBF 中心

正交最小二乘(Orthogonal least square)法是 RBF 神经网络的另一种重要的学习方法,其思想来源于线性回归模型。神经网络的输出实际上是隐含层神经元某种响应参数(这里称为回归因子)和隐含—输出层间连接权重的线性组合。所有隐含层神经元上的回归因子构成回归向量。正交最小二乘法的任务是通过神经网络的学习来获得合适的回归向量。学习过程主要是回归向量的正交化的过程。

实际应用表明,这些学习算法均有不足之处,使之应用范围受到限制。主要缺点体现在:如果隐含层神经元的取值是训练样本中的数据,那么在多数情况下难以反应系统的真正映射关系,并且在中心点的优选中会出现病态现象,导致训练失败。在很多实际问题中,RBF 神经网络隐含层神经元的中心并非是训练集中的某些样本点或样本的聚类中心,需要通过学习的方法获得,使所得到的中心能够更好地反应训练集数据所包含的信息。因此,有监督学习选取 RBF 神经网络中心的学习算法是一般的形式。但是,这种算法也有其缺点,即如果中心选取不当,会导致学习不收敛。因此,针对这种学习算法,并结合高斯(Gaussian)核函数的特点,给出了一种新的 RBF 神经网络学习算法——基于高斯核的 RBF 神经网络。

2. 基于高斯核的 RBF 神经网络拓扑结构

RBF 神经网络的拓扑结构是一种三层前馈网络:输入层由信号源结点构成,仅起到数据信息的传递作用,对输入信息不进行任何变换。第二层为隐含层,结点数视需要而定。隐含层神经元的核函数(作用函数)为高斯函数,对输入信息进行空间映射变换。第三层为输出层,它对输入模式做出响应。输出层神经元的作用函数为线性函数,对隐含层神经元输出的信息进行线性加权后输出,作为整个神经网络的输出结果。基于高斯核的 RBF 神经网络的拓扑结构如图 6-16 所示。

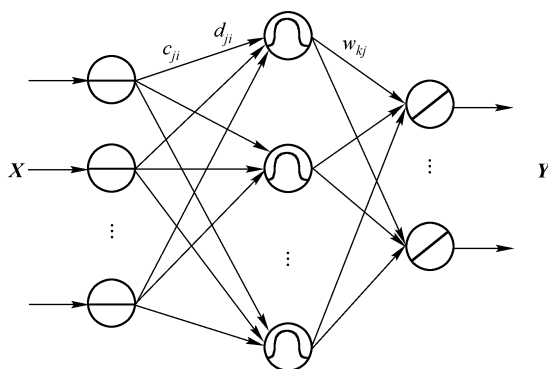


图 6-16 基于高斯核的 RBF 神经网络的拓扑结构

隐含层径向基神经元模型机构如图 6-17 所示。由图 6-17 可见,径向基网络传递函数是以输入向量与阈值向量之间的距离 $\|X - C_j\|$ 作为自变量的,其中 $\|X - C_j\|$ 是通过输入向量和加权矩阵 C 的行向量的乘积得到的。径向基神经网络传递函数可以取多种形式,最常用的有下面三种:

① Gaussian 函数

$$\phi_i(t) = e^{-\frac{t^2}{\delta_i^2}}$$

② Reflected sigmoidal 函数

$$\phi_i(t) = \frac{1}{1 + e^{\frac{t^2}{\delta_i^2}}}$$

③ 逆 Multiquadric 函数

$$\phi_i(t) = \frac{1}{(t^2 + \delta_i^2)^a} \quad a > 0$$

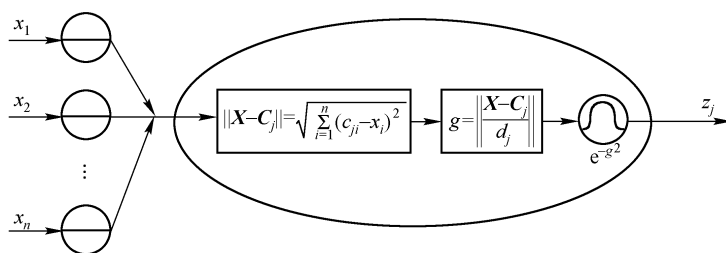


图 6-17 径向基神经元模型机构

但是,较为常用的还是 Gaussian 函数。本书选用 Gaussian 函数: $y = e^{-x^2}$ 为径向基函数。

当输入自变量为 0 时,传递函数取得最大值为 1。随着权值和输入向量间的距离不断减小,网络输出是递增的。也就是说,径向基函数对输入信号在局部产生响应。函数的输入信号 X 靠近函数的中央范围时,隐含层结点将产生较大的输出,如图 6-18 所示。由此可以看出这种网络具有局部逼近的能力。

当输入向量加到网络输入端时,径向基层每个神经元都会输出一个值,代表输入向量与神

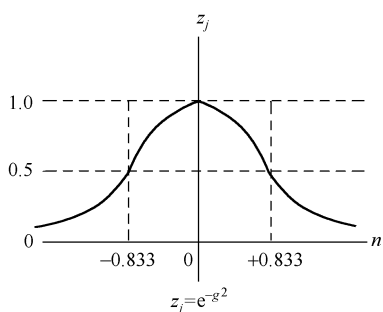


图 6-18 Gaussian 函数

神经元权值向量之间的接近程度。如果输入向量与权值向量相差很多,则径向基层输出接近于 0,经过第二层的线性神经元,输出也接近于 0;如果输入向量与权值向量很接近,则径向基层的输出接近于 1,经过第二层的线性神经元,输出值就靠近第二层权值。在这个过程中,如果只有一个径向基神经元的输出为 1,而其他神经元的输出均为 0 或者接近 0,那么线性神经元的输出就相当于输出为 1 的神经元相对应的第二层权值的值。一般情况下,不止一个径向基神经元的输出为 1,所以输出值也会有所不同。

3. RBF 网络训练

训练的目的是求两层的最终权值 C_j 、 D_j 和 W_j 。RBF 网络的训练过程分为两步:第一步为无导师学习,训练确定输入层与隐含层间的权值 C_j 、 D_j ;第二步为有导师学习,训练确定隐含层与输出层间的权值 W_j 。在训练前,需要提供输入向量 X 、对应的目标输出向量 Y 和径向基函数的宽度向量 D_j 。在第 l 次输入样品 ($l=1,2,\dots,N$) 进行训练时各个参数的表达及计算方法如下。

(1) 确定参数

① 确定输入向量 X :

输入向量为 $X = [x_1, x_2, \dots, x_n]^T$ (n ——输入层单元个数)。

② 确定输出向量 Y 和希望输出向量 O

输出向量为 $Y = [y_1, y_2, \dots, y_q]^T$ (q ——输出层单元数)。

希望输出向量 $O = [o_1, o_2, \dots, o_q]^T$ 。

③ 初始化隐含层至输出层的连接权值 $W_k = [w_{k1}, w_{k2}, \dots, w_{kp}]^T$ ($k=1,2,\dots,q$)。

参考中心初始化的方法给出:

$$W_{kj} = \min k + j \frac{\max k - \min k}{q + 1} \quad (6-24)$$

式中, $\min k$ 为训练集中第 k 个输出神经元所有期望输出的最小值; $\max k$ 为训练集中第 k 个输出神经元所有期望输出的最大值。

④ 初始化隐含层各神经元的中心参数 $C_j = [c_{j1}, c_{j2}, \dots, c_{jn}]^T$ 。不同隐含层神经元的中心应有不同的取值,并且与中心的对应宽度能够调节,使得不同的输入信息特征能被不同的隐含层神经元最大程度地反映出来,在实际应用时,一个输入信息总是包含在一定的取值范围内。不失一般性,将隐含层各神经元的中心分量的初值,按从小到大等间距变化,使较弱的输入信息在较小的中心附近产生较强的响应。间距的大小可由隐含层神经元的个数来调节。这样做的好处,能够通过试凑的方法找出较为合理的隐含层神经元个数,并使中心的初始化尽量合理,不同的输入特征更为明显地在不同的中心处反映出来,体现高斯核的特点。

基于上述思想,RBF 神经网络中心参数的初始值可由下式给出:

$$c_{ji} = \min i + \frac{\max i - \min i}{2p} + (j-1) \frac{\max i - \min i}{p} \quad (p \text{ 为隐含层神经元总个数}, j=1,2,\dots,p) \quad (6-25)$$

式中, \min_i 为训练集中第 i 个特征所有输入信息的最小值; \max_i 为训练集中第 i 个特征所有输入信息的最大值。

⑤ 初始化宽度向量 $\mathbf{D}_j = [d_{j1}, d_{j2}, \dots, d_{jn}]^T$ 。宽度向量影响着神经元对输入信息的作用范围: 宽度越小, 相应隐含层神经元作用函数的形状越窄, 那么处于其他神经元中心附近的信息在该神经元处的响应就越小。一般计算方法如下:

$$d_{ji} = d_f \sqrt{\frac{1}{N} \sum_{k=1}^N (x_i^k - c_{ji})^2} \quad (6-26)$$

式中, d_f 为宽度调节系数, 取值应小于 1, 作用是使每个隐含层神经元更容易实现对局部信息的感受能力, 有利于提高 RBF 神经网络的局部响应能力。

(2) 计算隐含层第 j 个神经元的输出值 z_j

$$z_j = \exp\left(-\left\|\frac{\mathbf{X} - \mathbf{C}_j}{D_j}\right\|^2\right) \quad j = 1, 2, \dots, p \quad (6-27)$$

式中, \mathbf{C}_j 为隐含层第 j 个神经元的中心向量, 由隐含层第 j 个神经元对应于输入层所有神经元的中心分量构成, $\mathbf{C}_j = [c_{j1}, c_{j2}, \dots, c_{jn}]^T$; D_j 为隐含层第 j 个神经元的宽度向量, 与 \mathbf{C}_j 相对应, $\mathbf{D}_j = [d_{j1}, d_{j2}, \dots, d_{jn}]^T$, D_j 越大, 隐含层对输入向量的响应范围就越大, 且神经元间的平滑度也较好; $\|\cdot\|$ 为欧式范数。

(3) 计算输出层神经元的输出

$$\mathbf{Y} = [y_1, y_2, \dots, y_q]^T$$

$$y_k = \sum_{j=1}^p w_{kj} z_j \quad k = 1, 2, \dots, q \quad (6-28)$$

式中, w_{kj} 为输出层第 k 个神经元与隐含层第 j 个神经元间的调节权重。

(4) 权重参数的迭代计算

RBF 神经网络权重参数的训练方法在这里取为梯度下降法。中心、宽度和调节权重参数均通过学习来自适应调节到最佳值, 它们的迭代计算如下:

$$w_{kj}(t) = w_{kj}(t-1) - \eta \frac{\partial E}{\partial w_{kj}(t-1)} + \alpha [w_{kj}(t-1) - w_{kj}(t-2)] \quad (6-29)$$

$$c_{ji}(t) = c_{ji}(t-1) - \eta \frac{\partial E}{\partial c_{ji}(t-1)} + \alpha [c_{ji}(t-1) - c_{ji}(t-2)] \quad (6-30)$$

$$d_{ji}(t) = d_{ji}(t-1) - \eta \frac{\partial E}{\partial d_{ji}(t-1)} + \alpha [d_{ji}(t-1) - d_{ji}(t-2)] \quad (6-31)$$

式中, $w_{kj}(t)$ 为第 k 个输出神经元与第 j 个隐层神经元之间在第 t 次迭代计算时的调节权重; $c_{ji}(t)$ 为第 j 个隐层神经元对应于第 i 个输入神经元在第 t 次迭代计算时的中心分量; $d_{ji}(t)$ 为与中心 $c_{ji}(t)$ 对应的宽度; η 为学习因子; E 为 RBF 神经网络评价函数, 由下式给出:

$$E = \frac{1}{2} \sum_{l=1}^N \sum_{k=1}^q (y_{lk} - O_{lk})^2 \quad (6-32)$$

式中, O_{lk} 为第 k 个输出神经元在第 l 个输入样本时的期望输出值; y_{lk} 为第 k 个输出神经元在第 l 个输入样本时的网络输出值。

综上所述, 可给出 RBF 神经网络如下的学习算法:

① 按式(6-24)~式(6-26)对神经网络参数进行初始化,并给定 η 和 α 的取值及迭代终止精度 ε 的值。

② 按下式计算网络输出的均方根误差 RMS 的值,若 $\text{RMS} \leq \varepsilon$,则训练结束,否则转到第③步。

$$\text{RMS} = \sqrt{\frac{\sum_{l=1}^N \sum_{k=1}^q (O_{lk} - y_{lk})^2}{qN}} \quad (6-33)$$

③ 按式(6-29)~式(6-31)对调节权重、中心和宽度参数进行迭代计算。

④ 返回步骤②。

6.3.2 径向基函数神经网络分类器设计

1. 实现步骤

① 从样本库中获取训练样本。

② 设置目标向量及径向基函数的分布密度。

③ 调用 newrbf,构建并训练径向基函数神经网络。newrbf 定义为: $\text{net} = \text{newrbf}(\mathbf{P}, \mathbf{T}, \text{spread})$,其中 \mathbf{P} 为输入向量, \mathbf{T} 为输出向量, spread 为径向基函数分布密度(默认值为 1)。

④ 获取手写数字特征,调用 sim,识别手写数字所属类别。

2. 编程代码

```
%%%
% 函数名称:rbfnet
% 函数功能:构建并训练 RBF 网络
% 函数参数:无
% 函数返回值:无
%%
function rbfnet;
    global rbfnet;
    clc;
    load templet pattern;
    c=0;
    for i=1:10
        for j=1:100
            c=c+1;
            p(:,c)=pattern(i).feature(:,j);
        end
    end
    tc(:,1:100)=0;
    tc(:,101:200)=1;
    tc(:,201:300)=2;
```

```

tc( :,301:400) = 3;
tc( :,401:500) = 4;
tc( :,501:600) = 5;
tc( :,601:700) = 6;
tc( :,701:800) = 7;
tc( :,801:900) = 8;
tc( :,901:1000) = 9;
tc = tc/10; % tc = [ 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2... ]
t = tc;
SPREAD = 1;
rbfnet = newrbf( p,t,SPREAD);
% 函数名称:rbfnettest
% 函数功能:识别手写数字
% 函数参数:手写数字特征 sample
% 函数返回值:手写数字所属类别 y
function y = rbfnettest( sample);
    global rbfnet;
    t = sim( rbfnet,sample');
    t = t * 10;
    y = round( t );

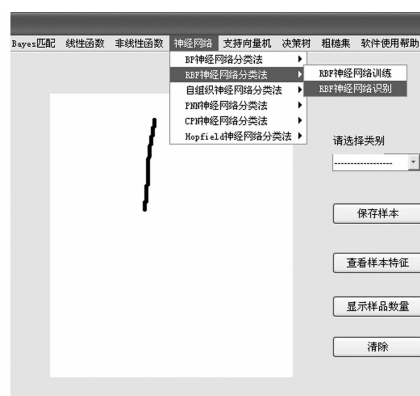
```

3. 效果图

① 选择“神经网络”→“RBF 神经网络分类法”→“RBF 神经网络训练”菜单命令,建立并训练 RBF 网络,如图 6-19 所示。



(a) 训练菜单



(b) 识别菜单

图 6-19 RBF 神经网络训练

② 拖动鼠标左键在视图区用鼠标手写一个数字,如图 6-20 所示,然后选择“神经网络”→“RBF 神经网络分类法”→“RBF 神经网络识别”菜单命令,进行手写数字分类。



图 6-20 RBF 神经网络识别

6.4 自组织竞争神经网络

在生物神经系统中,存在着一种“侧抑制”现象,即一个神经细胞兴奋后,通过它的分支会对周围其他神经细胞产生抑制。由于侧抑制的作用,各个细胞之间相互竞争的最终结果是:兴奋作用最强的神经元细胞所产生的抑制作用战胜了周围其他所有细胞的抑制作用而“赢”了,其周围的其他神经细胞全“输”了。

自组织竞争人工神经网络正是基于上述生物结构和现象形成的。它是一种以无导师学习方式对网络训练的,具有自组织能力的神经网络。它能够对输入模式进行自组织训练和判断,并将其最终分为不同的类型。与 BP 神经网络相比,这种自组织自适应的学习能力进一步拓宽了人工神经网络在模式识别、分类方面的应用,另外,竞争学习网络的核心——竞争层,又是许多种其他神经网络模型的重要组成部分。

在网络结构上,自组织竞争人工神经网络一般是由输入层和竞争层构成的两层网络,网络没有隐含层,输入层和竞争层之间的神经元实现双向连接,同时竞争层各个神经元之间还存在横向连接。在学习算法上,它模拟生物神经系统依靠神经元之间兴奋、协调与抑制、竞争的作用来进行信号处理的动力学原理,指导网络的学习与工作。

自组织竞争人工神经网络的基本思想是网络竞争层各个神经元竞争对输入模式的响应机会,最后仅有一个神经元成为竞争的获胜者,并对那些与获胜神经元有关的各个连接权值朝向更有利于竞争的方向调整,获胜神经元表示输入模式的分类。除了竞争方法外,还可以通过另一种手段获胜,即网络竞争层各神经元都能抑制所有其他神经元对输入模式的响应机会,从而使自己成为获胜者。此外,还有一种抑制的方法,即每个神经元只抑制与自己邻近的神经元,而对远离自己的神经元则不抑制。因此,自组织竞争神经网络具有自组织自适应的学习能力,进一步拓宽了神经网络在模式识别、分类方面的应用。

6.4.1 自组织竞争神经网络的基本概念

1. 自组织竞争神经网络学习规则

竞争网络在经过竞争而求得获胜结点后,则对与获胜结点相连的权值进行调整,调整权值的目的是为了使权值与其输入矢量之间的差别越来越小,从而使训练后的竞争网络的权值能够代表对应输入矢量的特征,把相似的输入矢量分成了同一类,并由输出来指示所代表的类别。竞争网络修正权值的公式为

$$\Delta w_{ij} = \alpha \cdot (x_i - w_{ij}) \quad (6-34)$$

式中, α 为学习速率,且 $0 < \alpha < 1$,一般的取值范围为 $0.01 \sim 0.3$; x_i 为经过归一化处理后的输入。

2. 自组织神经网络的拓扑结构

自组织神经网络是一类无导师学习的神经网络模型,这类模型大都采用竞争型学习规则,可以对外界未知环境(或样本空间)进行学习或仿真,并对自身的网络结构进行适当调整。竞争型神经网络可分为输入层和竞争层,自组织神经网络结构如图6-21所示。

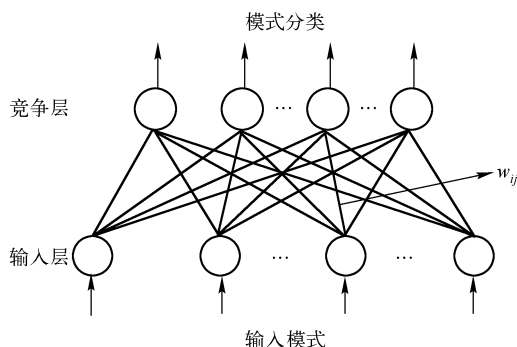


图 6-21 自组织竞争神经网络结构

3. 自组织竞争网络训练

竞争网络训练实际上是对输入矢量的划分聚类过程,使得获胜结点与输入矢量之间的权矢量代表获胜输入矢量。这样,当达到最大循环的值后,网络已重复多次训练了训练模式 \mathbf{X} 中的所有矢量,训练结束后,对于用于训练的模式 \mathbf{X} ,其网络输出矢量中,其值为1的代表一种类型,而每类的典型模式值由该输出结点与输入结点相连的权矢量表示。在第 l 次输入样品 ($l=1,2,\dots,N$) 进行训练时各个参数的表达及计算方法如下。

(1) 确定参数

① 确定竞争网络的输入层结点。竞争网络的输入层结点是由已知输入矢量决定的。

$\mathbf{X} = [x_1, x_2, \dots, x_n]$ (n 为输入层单元个数)。

输入样本为二值向量,各个元素的取值是0或1。

② 确定竞争层的神经元数 p 。竞争层的神经元数 p 是由设计者确定的,一般情况下,可以根据输入矢量的维数及其估计,再适当地增加些数目来确定。

③ 确定学习速率和最大循环次数。竞争网络的训练是在达到最大循环次数后停止,这个数一般可取输入矢量数组的 15 ~ 20 倍,即使每组输入矢量能够在网络重复出现 15 ~ 20 次。通过重复训练,自组织竞争网络将所有输入向量进行了分类。

④ 确定输入层到竞争层的权值 $\mathbf{W}_j = [w_{j1}, w_{j2}, \dots, w_{jn}]^T$ 。网络的连接权值为 w_{ji} , $i = 1, 2, \dots, n; j = 1, 2, \dots, p$, 且满足约束条件 $\sum_{i=1}^n w_{ji} = 1$ 。竞争网络的权值要进行随机归一化的初始化处理,然后网络则可以进入竞争以及权值的调整阶段。

(2) 计算竞争层神经元 j 的状态 s_j

竞争层神经元 j 的状态可按下式计算:

$$s_j = \sum_{i=1}^n w_{ji} x_i \quad (6-35)$$

式中, x_i 为输入样本向量的第 i 个元素。

(3) 求解赢得竞争胜利的神经元

竞争胜利的神经元就代表着当前输入样本的分类模式。根据竞争机制,竞争层中具有最大加权值的神经元 k 赢得竞争胜利,输出为

$$a_k = \begin{cases} 1, & s_k > s_j, \forall j, k \neq j \\ 0, & \text{其他} \end{cases} \quad (6-36)$$

(4) 竞争后获胜结点权值修正

在竞争层中,神经元之间相互竞争,最终只有一个或者几个神经元获胜,以适应当前的输入样本。只有与获胜结点相连的权值才能得到修正,并且通过其学习法则修正后的权值更加接近其获胜输入向量。竞争后获胜结点的权值按照下式进行修正:

$$w_{ji} = w_{ji} + \alpha \left(\frac{x_i}{m} - w_{ji} \right) \quad (6-37)$$

式中, α 为学习参数, $0 < \alpha < 1$, 一般取为 0.01 ~ 0.03; m 为输入层中输出为 1 的神经元个数,即

$$m = \sum_{i=1}^n x_i。$$

权值调整式中的 $\frac{x_i}{m}$ 项表示当 x_i 为 1 时,权值增加;而当 x_i 为 0 时,权值减小。也就是说,当 x_i 活跃时,对应的第 i 个权值就增加,否则就减小。由于所有权值的和为 1,所以当第 i 个权值增加或减小时,对应的其他权值就可能减小或增加。此外,该式还保证了权值的调整能够满足所有的权值调整量之和为 0。

获胜的结点对将来再次出现的相似向量更加容易赢得该结点的胜利。而对于一个不同的向量出现时,就更加不易取胜,但可能是其他某个结点获胜,归于另一类向量群中。随着输入向量的不断出现而不断调整获胜者相连的权向量,以使其更加接近于某一类输入向量。最终,如果有足够的神经元结点,每一组输入向量都能使某一结点的输出为 1 而聚为此类。

6.4.2 自组织竞争神经网络分类器设计

1. 实现步骤

- ① 提取每类的所有样本的均值作为该类的代表,组成训练样本矩阵。
- ② 调用 `newc`,构建自组织神经网络。
- ③ 调用 `train`,训练网络。
- ④ 调用 `sim`,对手写数字进行仿真识别,将识别出的结果与每类代表该次训练所属的类号进行比对,确定识别结果。

2. 编程代码

```
%%%%%%%%%%%%%%
% 函数名称:zizuzhitrain
% 函数功能:构建并训练自组织网络
% 函数参数:无
% 函数返回值:无
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function zizuzhitrain
    global net;
    global T;
    T=[0 0 0 0 0 0 0 0 0]; % 存储每次训练后训练集每类所属的类号(各次训练每类所属的类号不同)
    load templet pattern;
    % 取得第一类所有样本的平均特征向量
    a = pattern(1,1).feature(:,1:130);
    b = cumsum(a,2);
    c = b(:,130);
    d = c/130;
    A = d;
    for i = 2:10
        ax = pattern(1,i).feature(:,1:130);
        bx = cumsum(ax,2);
        cx = bx(:,130);
        dx = cx/130;
        B = dx;% 取得各类各自所有样本的平均特征向量
        C = [A B];% 矩阵拼接
        A = C;
    end
    net = newc(minmax(C),10,0.1);% 构建自组织网络
    net.trainParam.epochs = 400;% 训练次数
    net = train(net,C); % 训练网络
```



```

Y = sim(net,C) % 训练集分类结果
T = vec2ind(Y) % 训练集分类结果类别
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 函数名称:zizuzhi
% 函数功能:使用 zizuzhi 网络,识别手写数字
% 函数参数:手写数字特征 sample
% 函数返回值:手写数字所属类别 y
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = zizuzhi(sample);
    clc;
    global net;
    global T;% 存储每次训练后训练集每类所属的类号(各次训练每类所属的类号不同)
    yt = sim(net,sample');
    yy = vec2ind(yt);
    switch(yy)% 翻译识别结果
        case T(1,1)
            y = 0;
            return;
        case T(1,2)
            y = 1;
            return;
        case T(1,3)
            y = 2;
            return;
        case T(1,4)
            y = 3;
            return;
        case T(1,5)
            y = 4;
            return;
        case T(1,6)
            y = 5;
            return;
        case T(1,7)
            y = 6;
            return;
        case T(1,8)
            y = 7;
            return;
        case T(1,9)
            y = 8;
            return;
        case T(1,10)

```

```
y = 9;  
return;  
  
end  
y = -1;
```

3. 效果图

① 选择“神经网络”→“自组织神经网络分类法”→“自组织神经网络训练”菜单命令,建立并训练网络,如图 6-22 所示。

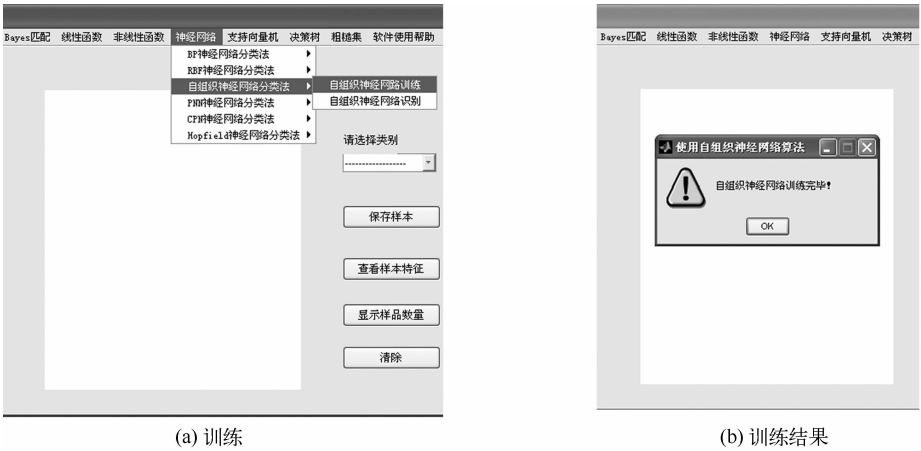


图 6-22 自组织神经网络训练

- ② 训练后权值(详见“自组织网络训练后的权值和阈值.txt”)。
- ③ 拖动鼠标左键在视图区用鼠标手写一个数字,如图 6-23 所示,然后选择“神经网络”→“自组织神经网络分类法”→“自组织神经网络识别”菜单命令,进行手写数字分类。

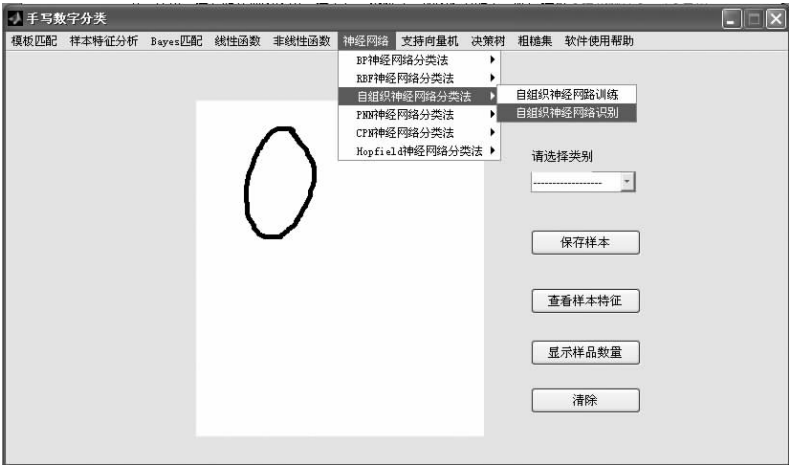


图 6-23 自组织神经网络分类

自组织手写数字的识别效果如图 6-24 所示。



图 6-24 自组织手写数字的识别效果

6.5 概率神经网络(PNN)

6.5.1 概率神经网络的基本概念

概率神经网络(Probabilistic Neural Networks, PNN)是由 D. F. Specht[1]在 1990 年提出的。主要思想是用贝叶斯决策规则,即错误分类的期望风险最小,在多维输入空间内分离决策空间。它是一种基于统计原理的人工神经网络,它是以 Parzen 窗口函数为激活函数的一种前馈网络模型。PNN 吸收了径向基神经网络与经典的概率密度估计原理的优点,与传统的前馈神经网络相比,径向基神经元还可以和竞争神经元一起共同组建概率神经网络(Probabilistic Neural Network, PNN)。在模式分类方面尤其具有较为显著的优势。

1. 概率神经网络拓扑结构

概率神经网络由四层结构组成,如图 6-25 所示。首先输入层,进行待测样品向量输入;第二层计算输入向量与训练样本之间的距离,表示输入向量与训练样本之间的接近程度;第三层将与输入向量相关的所有类别综合在一起,网络输出为表示概率的向量;最后通过第四层的竞争(Compete)传递函数进行取舍,概率最大值的那一类为 1,其他类别用 0 表示。

输入层:首先将输入向量 \bar{x} 输入到输入层,网络计算输入向量与训练样本向量之间的差值 $\bar{x} - \bar{x}_{ik}$,差值绝对值 $\|\bar{x} - \bar{x}_{ik}\|$ 的大小代表这两个向量之间的距离,所得的向量由输入层输出,该向量反映了向量间的接近程度;接着,把输入层的输出向量 $\bar{x} - \bar{x}_{ik}$ 送入到样本层中。

样本层:样本层结点的数目等于训练样本数目的总和, $N = \sum_{i=1}^M N_i$, 其中 M 是类的总数。先判断哪些类别与输入向量有关,再将相关度高的类别集中起来,样本层的输出值就代表相识度;然后,将样本层的输出值送入到求和层。

求和层:求和层的结点个数是 M ,每个结点对应一个类,通过求和层的竞争传递函数进行判决。

竞争层:最后判决的结果由竞争层输出,输出结果中只有一个 1,其余结果都是 0,概率值

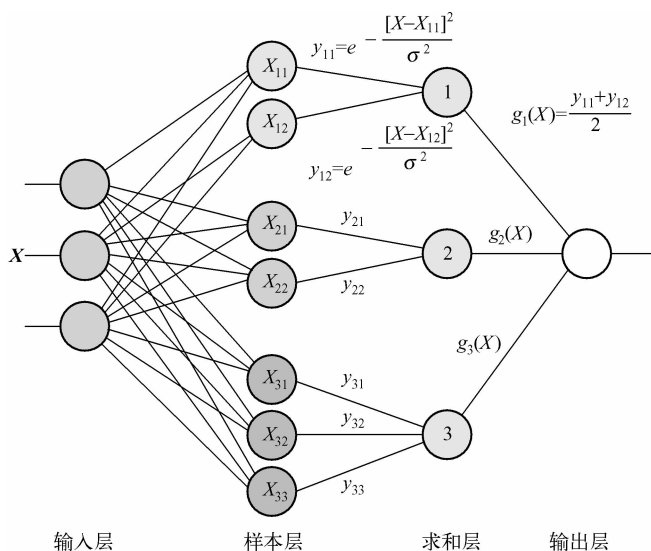


图 6-25 概率神经网络拓扑结构图

最大的那一类输出结果为 1。

2. 概率神经网络的工作过程

(1) 确定参数

① 确定输入层,有 n 个神经元, p 个待测样品,每个样品有 n 个特征。

输入模式为 $\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ d_{p1} & d_{p2} & \cdots & d_{pn} \end{bmatrix}$, 将输入向量 \mathbf{D} 进行归一化处理:

$$d_i = \frac{d}{\|\mathbf{D}\|}, \|\mathbf{D}\| = \sqrt{\sum_{i=1}^n (d_i)^2}, (i=1, 2, \cdots, n)$$

② 确定样本层,原始学习样本有 m 个,就有 m 个神经元。

归一化学习矩阵 \mathbf{C} 。样本层结点个数为训练样本个数,设原始学习样本有 m 个,每一个样本的特征属性有 n 个。对样本层矩阵进行归一化处理,可以减小误差,避免较小的值被较大的值“吃掉”。训练学习样本矩阵为

$$\mathbf{X} = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1n} \\ X_{21} & X_{22} & \cdots & X_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ X_{m1} & X_{m2} & \cdots & X_{mn} \end{bmatrix} \quad (6-38)$$

从样本的矩阵式(6-38)中可以看出,在求归一化因子之前,必须先计算 \mathbf{B}^T 矩阵

$$\mathbf{B}^T = \left[\frac{1}{\sqrt{\sum_{k=1}^n x_{1k}^2}} \quad \frac{1}{\sqrt{\sum_{k=1}^n x_{2k}^2}} \quad \cdots \quad \frac{1}{\sqrt{\sum_{k=1}^n x_{mk}^2}} \right]$$

然后计算

$$C_{m \times n} = B_{m \times 1} [1 \quad 1 \quad \cdots \quad 1]_{1 \times n} \cdot X_{m \times n} = \begin{bmatrix} \frac{x_{11}}{\sqrt{M_1}} & \frac{x_{12}}{\sqrt{M_1}} & \cdots & \frac{x_{1n}}{\sqrt{M_1}} \\ \frac{x_{21}}{\sqrt{M_2}} & \frac{x_{22}}{\sqrt{M_2}} & \cdots & \frac{x_{2n}}{\sqrt{M_2}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{x_{m1}}{\sqrt{M_m}} & \frac{x_{m2}}{\sqrt{M_m}} & \cdots & \frac{x_{mn}}{\sqrt{M_m}} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1n} \\ C_{21} & C_{22} & \cdots & C_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ C_{m1} & C_{m2} & \cdots & C_{mn} \end{bmatrix} \quad (6-39)$$

式中, $M_1 = \sum_{k=1}^n x_{1k}^2, M_2 = \sum_{k=1}^n x_{2k}^2, \cdots, M_m = \sum_{k=1}^n x_{mk}^2$

在式(6-39)中,符号“ \cdot ”表示矩阵在做乘法运算时,相应元素之间的乘积。

因为是有监督的学习算法,所以很容易就知道每个样本属于那种类型。假设 m 个样本一共可以分为 c 类,并且各类样本的数目相同,设为 k ,于是 $m = k * c$ 。

③ 确定求和层,有 c 个类别,就有 c 个神经元。每个结点对应一个类输出, $Y = [y_1, y_2, \cdots, y_c]^T$ 。

(2) 模式距离的计算

该距离是指待测样本矩阵与学习矩阵中相应元素之间的距离。假设将由 p 个 n 维向量组成的矩阵称为待识别样本矩阵,则经归一化后,需要待识别的输入样本矩阵为

$$D = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ d_{p1} & d_{p2} & \cdots & d_{pn} \end{bmatrix} \quad (6-40)$$

计算欧式距离,就是需要计算每个待测样品到训练集中已经识别样本的距离。

$$E = \begin{bmatrix} \sqrt{\sum_{k=1}^n |d_{1k} - c_{1k}|^2} & \sqrt{\sum_{k=1}^n |d_{1k} - c_{2k}|^2} & \cdots & \sqrt{\sum_{k=1}^n |d_{1k} - c_{mk}|^2} \\ \sqrt{\sum_{k=1}^n |d_{2k} - c_{1k}|^2} & \sqrt{\sum_{k=1}^n |d_{2k} - c_{2k}|^2} & \cdots & \sqrt{\sum_{k=1}^n |d_{2k} - c_{mk}|^2} \\ \cdots & \cdots & \cdots & \cdots \\ \sqrt{\sum_{k=1}^n |d_{pk} - c_{1k}|^2} & \sqrt{\sum_{k=1}^n |d_{pk} - c_{2k}|^2} & \cdots & \sqrt{\sum_{k=1}^n |d_{pk} - c_{mk}|^2} \end{bmatrix} = \begin{bmatrix} E_{11} & E_{12} & \cdots & E_{1m} \\ E_{21} & E_{22} & \cdots & E_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ E_{p1} & E_{p2} & \cdots & E_{pm} \end{bmatrix} \quad (6-41)$$

(3) 激活样本层径向基函数的神经元

学习样本 C 与待识别样本 D 被归一化后,通常取标准差 $\sigma = 0.1$ 的高斯型函数。激活后

得到初始概率矩阵

$$P = \begin{bmatrix} e^{-\frac{E_{11}}{2\sigma^2}} & e^{-\frac{E_{12}}{2\sigma^2}} & \cdots & e^{-\frac{E_{1m}}{2\sigma^2}} \\ e^{-\frac{E_{21}}{2\sigma^2}} & e^{-\frac{E_{22}}{2\sigma^2}} & \cdots & e^{-\frac{E_{2m}}{2\sigma^2}} \\ \cdots & \cdots & \cdots & \cdots \\ e^{-\frac{E_{p1}}{2\sigma^2}} & e^{-\frac{E_{p2}}{2\sigma^2}} & \cdots & e^{-\frac{E_{pm}}{2\sigma^2}} \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1m} \\ P_{21} & P_{22} & \cdots & P_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ P_{p1} & P_{p2} & \cdots & P_{pm} \end{bmatrix} \quad (6-42)$$

(4) 求和层计算各个样本属于各类的初始概率和

假设样本有 m 个,那么一共可以分为 c 类,并且各类样本的数目相同,设为 k ,则可以在网络的求和层求得各个样本属于各类的初始概率和

$$S = \begin{bmatrix} \sum_{l=1}^k P_{1l} & \sum_{l=k+1}^{2k} P_{1l} & \cdots & \sum_{l=m-k+1}^m P_{1l} \\ \sum_{l=1}^k P_{2l} & \sum_{l=k+1}^{2k} P_{2l} & \cdots & \sum_{l=m-k+1}^m P_{2l} \\ \cdots & \cdots & \cdots & \cdots \\ \sum_{l=1}^k P_{pl} & \sum_{l=k+1}^{2k} P_{pl} & \cdots & \sum_{l=m-k+1}^m P_{pl} \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1c} \\ S_{21} & S_{22} & \cdots & S_{2c} \\ \cdots & \cdots & \cdots & \cdots \\ S_{p1} & S_{p2} & \cdots & S_{pc} \end{bmatrix} \quad (6-43)$$

在上式中, S_{ij} 代表的意思是:将要被识别的样本中,第 i 个样本属于第 j 类的初始概率和。

(5) 竞争层

通过计算概率 $prob_{ij}$,即第 i 个样本属于第 j 类的概率,找出每行中最大的概率,求得每个样品的类别。

$$prob_{ij} = \frac{S_{ij}}{\sum_{l=1}^c S_{il}} \quad (6-44)$$

6.5.2 概率神经网络分类器设计

设隐含层中心向量数目为 p ,期望值为 M ,表示类别只有一个元素为 1,其余均为 0。

PNN 网络第一层的输入权值 C 为隐含层神经元中心向量,经过距离计算后,第一层输入向量表示输入向量与训练样本向量的接近程度,然后与阈值向量相除,再经过径向传递函数计算。输入向量与哪个输入样本最接近,则神经元输出 Z 对应元素就为 1,如果输入向量与几个类别的输入样本都接近,则 Z 相对应的几个元素均为 1。

第二层权值矩阵 V ,每个行向量只有一个元素为 1,代表相应的类别,其余元素为 0,然后计算乘积 VZ 。最后通过第二层传递函数竞争计算得到输出,较大的元素取值为 1,其余为 0。至此 PNN 网络就能够完成对输入向量的分类了。

概率神经网络按此方式进行分类,为网络提供一种输入模式向量后,首先,径向基层计算该输入向量与样本输入向量间的距离,该层输出为一个距离向量。竞争层接受距离向量为输入向量,计算每个模式出现的概率,通过竞争传递函数为概率最大的元素对应的输出为 1,这

就是一类模式;否则输出为 0,作为其他分类模式。

1. 实现步骤

- ① 提取样本库样品。
- ② 提取样本库样品所属的类别;调用 ind2vec 函数,将类别向量转换为 PNN 可以使用的目标向量。
- ③ 调用函数 rbfpnnttrain,构建并训练 PNN 网络。
- ④ 调用 sim 函数,对手写数字进行仿真实验。
- ⑤ 调用 vec2ind 函数将分类结果转换为容易识别的类别向量。

2. 编程代码

```
%%%%%%%%%%%%%%% 函数名称:rbfpnnttrain
% 函数功能:构建并训练 PNN 网络
% 函数参数:无
% 函数返回值:无
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function pnntrain
    global pnnnet;
    clc;
    load templet pattern;
    c = 0;
    for i = 1:10
        for j = 1:20
            c = c + 1;
            p(:,c) = pattern(i).feature(:,j);
        end
    end
    tc(:,1:20) = 1;
    tc(:,21:40) = 2;
    tc(:,41:60) = 3;
    tc(:,61:80) = 4;
    tc(:,81:100) = 5;
    tc(:,101:120) = 6;
    tc(:,121:140) = 7;
    tc(:,141:160) = 8;
    tc(:,161:180) = 9;
    tc(:,181:200) = 10;
    tc = tc;
    t = ind2vec(tc);
    pnnnet = newpnn(p,t); % 构建神经网络
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 函数名称:pnnnet
% 函数功能:识别手写数字
% 函数参数:手写数字特征:sample
```

```
% 函数返回值:手写数字所属类别 yc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function yc = pnnnet(sample)
    clc;
    global rbfnet;
    y = sim(pnnnet,sample'); % 测试
    yc = vec2ind(y) - 1;
```

3. 效果图

① 选择“神经网络”→“RBF 神经网络分类法”→“PNN 概率神经网络训练”菜单命令,建立并训练 PNN 网络,如图 6-26 所示。

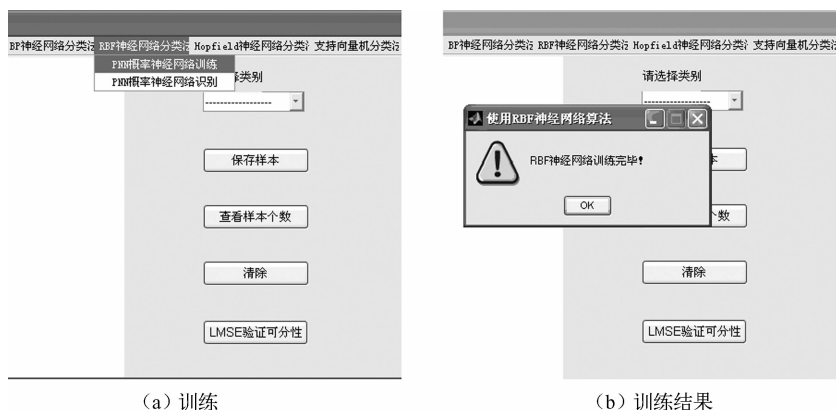


图 6-26 PNN 概率神经网络训练

② 训练后的权值详见文件《RBF 网络训练后的权值和阈值.txt》。

③ 拖动鼠标左键在视图区用鼠标手写一个数字,如图 6-27 所示,然后选择“神经网络”→“RBF 神经网络分类法”→“PNN 概率神经网络识别”菜单命令,进行手写数字分类。

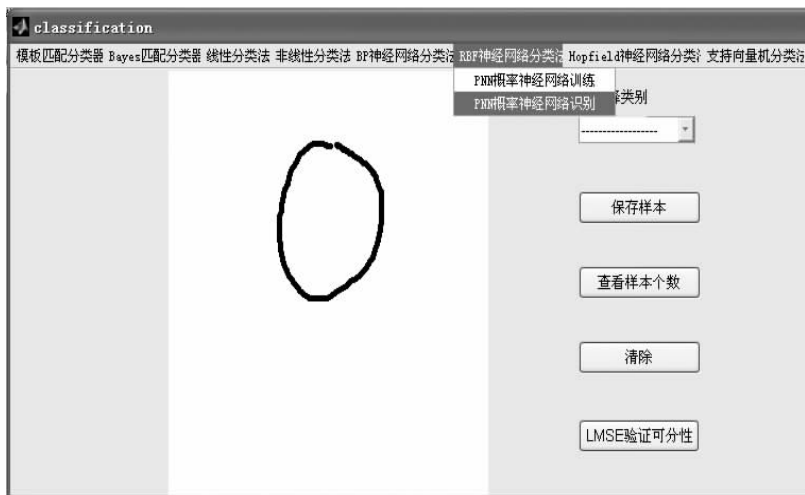


图 6-27 PNN 概率神经网络识别

PNN 神经网络手写数字的识别效果,如图 6-28 所示。



图 6-28 PNN 神经网络手写数字的识别效果

6.6 对向传播神经网络(CPN)

对向传播(Counter Propagation)网络,简称 CPN,是将 Kohonen 特征映射网络与 Grossberg 基本竞争型网络相结合,发挥各自的特长的一种新型特征映射网络。这一网络是美国计算机专家 Robert Hecht-Nielsen 于 1987 年提出的。这种网络被广泛应用于模式分类、函数近似、统计分析和数据压缩等领域。

6.6.1 对向传播神经网络的基本概念

CPN 网络结构如图 6-29 所示,网络分为输入层、竞争层和输出层。输入层与竞争层构成 SOM 网络,竞争层与输出层构成基本竞争型网络。从整体上看,网络属于有导师型的网络,而

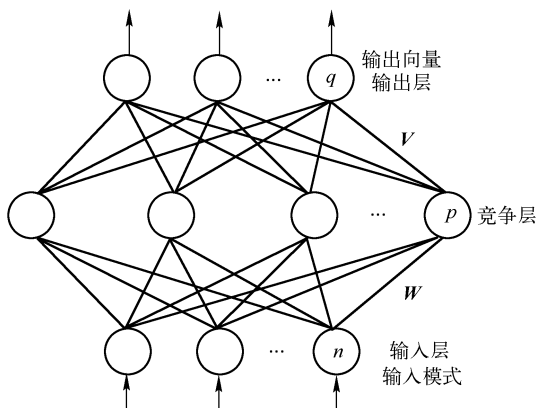


图 6-29 CPN 网络结构

由输入层和竞争层构成的 SOM 网络又是一种典型的无导师型的神经网络。因此,这一网络既涉及了无导师网络分类灵活、算法简练的优点,又采纳了有导师型网络分类精细、准确的长处,使两种不同类型的网络有机地结合起来。

CPN 网络的基本思想是,由输入层到竞争层,网络按照 SOM 学习规则产生竞争层的获胜神经元,并按照这一规则调整相应的输入层到竞争层的连接权;由竞争层到输出层,网络按照基本竞争型网络学习规则,得到各输出神经元的实际输出值,并按照有导师型

的误差校正方法,修正由竞争层到输出层的连接权。经过这样的反复学习,可以将任意的输入模式映射为输出模式。

从这一基本思想可以发现,处于网络中间未知的竞争层获胜神经元以及与其相关的连接权向量,既反映了输入模式的统计特性,又反映了输出模式的统计特性。因此,可以认为,输入、输出模式通过竞争层实现了相互映射,即网络具有双向记忆的性能。在第 l 次输入样品 ($l=1,2,\dots,N$) 进行训练时各个参数的表达及计算方法如下。

① 确定参数。

确定输入层有 n 个神经元。

输入模式为 $\mathbf{X} = [x_1, x_2, \dots, x_n]^T$, 将输入向量 \mathbf{X} 进行归一化处理:

$$x_i = \frac{x_i}{\|\mathbf{X}\|}, \|\mathbf{X}\| = \sqrt{\sum_{i=1}^n (x_i)^2}, (i = 1, 2, \dots, n) \quad (6-45)$$

确定竞争层有 p 个神经元。对应的二值输出向量为 $\mathbf{B} = [b_1, b_2, \dots, b_p]^T$ 。

确定输出层输出向量为 $\mathbf{Y} = [y_1, y_2, \dots, y_q]^T$, 目标输出向量为 $\mathbf{O} = [o_1, o_2, \dots, o_q]^T$ 。

确定由输入层到竞争层的连接权值向量为 $\mathbf{W}_j = [w_{j1}, w_{j2}, \dots, w_{jn}]^T, j = 1, 2, \dots, p$, 将连接权向量 \mathbf{W}_j 赋值为 $[0, 1]$ 内的随机值。

确定由竞争层到输出层的连接权向量为 $\mathbf{V}_k = [v_{k1}, v_{k2}, \dots, v_{kp}]^T, k = 1, 2, \dots, q$, 将连接权向量 \mathbf{V}_k 赋值为 $[0, 1]$ 内的随机值。

② 将连接权向量 \mathbf{W}_j 进行归一化处理:

$$w_{ji} = \frac{w_{ji}}{\|\mathbf{W}_j\|}, \|\mathbf{W}_j\| = \sqrt{\sum_{i=1}^n w_{ji}^2} \quad (6-46)$$

③ 求竞争层中每个神经元的加权输入和:

$$S_j = \sum_{i=1}^n x_i w_{ji}, j = 1, 2, \dots, p \quad (6-47)$$

④ 求连接权向量 \mathbf{W}_j 与 \mathbf{X} 距离最近的向量:

$$\mathbf{W}_g = \max_{j=1,2,\dots,p} \sum_{i=1}^n x_i w_{ji} = \max_{j=1,2,\dots,p} S_j \quad (6-48)$$

⑤ 将神经元 g 的输出设定为 1, 其余神经元输出设定为 0:

$$b_j = \begin{cases} 1, & j = g \\ 0, & j \neq g \end{cases} \quad (6-49)$$

⑥ 修正连接权向量 \mathbf{W}_g :

$$w_{gi}(t+1) = w_{gi}(t) + \alpha(x_i - w_{gi}(t)) \quad i = 1, 2, \dots, n; 0 < \alpha < 1 \quad (6-50)$$

⑦ 归一化连接权向量 \mathbf{W}_g :

$$w_{gi} = \frac{w_{gi}}{\|\mathbf{W}_g\|}, \|\mathbf{W}_g\| = \sqrt{\sum_{i=1}^n w_{gi}^2} \quad (6-51)$$

⑧ 求输出层各神经元的加权输出, 将其作为输出神经元的实际输出值:

$$y_k = \sum_{j=1}^p v_{kj} b_j \quad k = 1, 2, \dots, q \quad (6-52)$$

⑨ 只需调整竞争层中获胜神经元 g 到输出神经元的连接权向量 \mathbf{V}_g , 按照下式修正竞争层到输出层的连接权向量 \mathbf{V}_g :

$$v_{kg}(t+1) = v_{kg}(t) + \beta b_j (y_k - o_k) \quad k = 1, 2, \dots, q \quad (6-53)$$

式中, $0 < \beta < 1$ 为学习速率。

⑩ 返回②, 直到将 N 个输入模式全部提供给网络。

⑪ 令 $t = t + 1$, 将输入模式 \mathbf{X} 重新提供给网络学习, 直到 $t = T$ 。其中 T 为预先设定的学习总次数, 一般取为 $500 < T < 10000$ 。

6.6.2 对向传播神经网络分类器设计

1. 实现步骤

- ① 初始化输入层和竞争层、竞争层和输出层之间的连接权向量矩阵 \mathbf{W} 、 \mathbf{V} 。
- ② 取得训练样本, 并对其进行归一化处理。
- ③ 设定期望输出模式。
- ④ 归一化连接权向量 \mathbf{W} 。
- ⑤ 计算每一个竞争层神经元的加权输出 \mathbf{S}_j 。
- ⑥ 找出竞争层加权输出最大的神经元, 并将该神经元的输出设置为 1, 同层其他神经元的输出设置为 0。
- ⑦ 调整与竞争层输出为 1 的神经元相关的连接权向量, 并进行归一化。
- ⑧ 修正竞争层和输出层之间的连接权向量 \mathbf{V} 。
- ⑨ 计算输出层各神经元的加权输入, 并将其作为神经元的实际输出值 \mathbf{Y} 。
- ⑩ 对每一个训练样本, 均执行步骤④~⑨, 直至所有训练样本均经过一次训练。
- ⑪ 反复训练所有的训练样本, 直至指定的训练遍数。
- ⑫ 归一化手写数字样本, 计算每一个竞争层神经元的加权输出 \mathbf{S} 。
- ⑬ 找出竞争层加权输出最大的神经元, 并将该神经元的输出设置为 1, 同层其他神经元的输出设置为 0, 得到输出模式。
- ⑭ 将输出模式转换成十进制形式, 得到最终结果。

2. 编程代码

```
%%%%%%%%%%%%%%
% 函数名称:cpntrain
% 函数功能:构建 CPN 网络,使用梯度下降法训练 CPN 网络
% 函数参数:无
% 函数返回值:无
%%%%%%%%%%%%%
function cpntrain
    global w;
```

```

global v;
w = rands(250,25)/2 + 0.5;
v = rands(4,250)/2 + 0.5;
load templet pattern;
a = pattern(1,1).feature(:,1:130);
b = cumsum(a,2);
c = b(:,130);
d = c/130;
A = d;
for i = 2:10
    i;
    ax = pattern(1,i).feature(:,1:130);
    bx = cumsum(ax,2);
    cx = bx(:,130);
    dx = cx/130;
    B = dx;
    P = [A B];
    A = P;
end
T = [0 0 0 0;0 0 0 1;0 0 1 0;0 0 1 1;0 1 0 0;0 1 0 1;0 1 1 0;0 1 1 1;1 0 0 0;1 0 0 1];
T_out = T;
epoch = 1000;
for i = 1:10
    P(i,:) = P(i,+)/norm(P(i,:));
end
P = P';
while epoch > 0
    for j = 1:10
        for i = 1:250
            w(i,:) = w(i,+)/norm(w(i,:));
            s(i) = P(j,:) * w(i,)' ;
        end
        temp = max(s);
        for i = 1:250
            if temp == s(i)
                count = i;
            end
        end
        for i = 1:250
            s(i) = 0;
        end
        s(count) = 1;
        w(count,:) = w(count,:) + 0.1 * [P(j,:) - w(count,:)] ;
    end
    epoch = epoch - 1;
end

```

```

w(count,:) = w(count,:)/norm(w(count,:));
v(:,count) = v(:,count) + 0.1 * (T(j,:)' - T_out(j,:))';
T_out(j,:) = v(:,count)';

end
epoch = epoch - 1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 函数名称:cpn
% 函数功能:识别手写数字
% 函数参数:手写数字特征 sample
% 函数返回值:手写数字所属类别 y
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = cpn(sample);

global w;
global v;
sample(1,:) = sample(1,:)/norm(sample(1,:));
Outc = [0 0 0 0];
for i = 1:250
    sc(i) = sample(1,:) * w(i,:)' ;
end
tempc = max(sc);
for i = 1:250
    if tempc == sc(i)
        countp = i;
    end
    sc(i) = 0;
end
sc(countp) = 1;
Outc(1,:) = v(:,countp)';
Outca = round(Outc') % 转成最接近的二进制数
Outcb = num2str(Outca)
Outcc = bin2dec(Outcb')
y = Outcc

```

3. 效果图

① 选择“神经网络”→“CPN 神经网络分类法”→“CPN 神经网络训练”菜单命令,建立并训练 BP 神经网络,如图 6-30 所示。

② 训练后权值详见“CPN 网络训练后的权值和阈值.txt”。

③ 拖动鼠标左键在视图区用鼠标手写一个数字,如图 6-31 所示,然后选择“神经网络”→“对向传播神经网络分类”→“对向传播神经网络分类”菜单命令,进行手写数字分类。

CPN 手写数字的识别效果如图 6-32 所示。



(a) 训练



(b) 训练结果

图 6-30 对向传播(CPN)神经网络训练

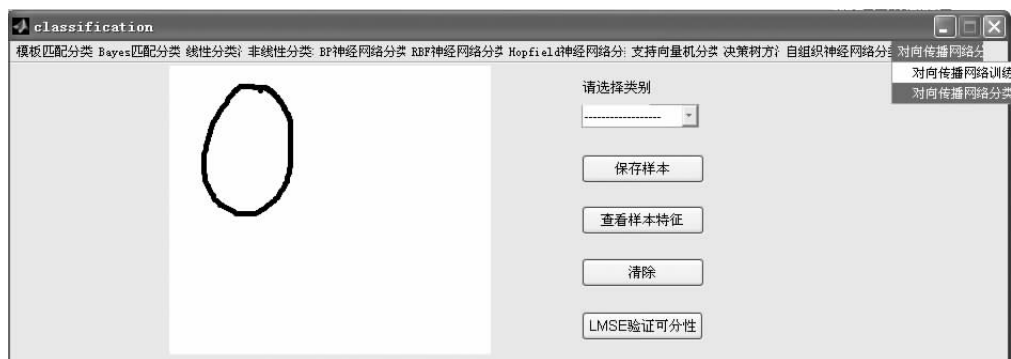


图 6-31 对向传播(CPN)神经网络识别



图 6-32 CPN 手写数字的识别效果

6.7 反馈型神经网络(Hopfield)

Hopfield 网络是最典型的反馈网络模型,它是目前人们研究最多的模型之一。Hopfield 网络是由相同的神经元构成的单层,并且具有学习功能的自联想网络,可以完成制约优化和联想记忆等功能。

6.7.1 Hopfield 网络的基本概念

1. Hopfield 网络简述

Hopfield 模型是霍普菲尔特(Hopfield)分别于1982年及1984年提出的。1984年,J. Hopfield 提出了可用做联想存储的互联网络,这个网络成为 Hopfield 网络模型,也成为 Hopfield 模型。Hopfield 神经网络模型是一种循环神经网络,从输出到输入有反馈连接。它有两个神经网络模型,一个是离散的,一个是连续的,但它们都属于反馈网络,即它们从输入层至输出层都有反馈存在。

反馈神经网络由于其输出端有反馈到其输入端,所以,Hopfield 网络在输入的激励下,会产生不断的状态变化。当有输入之后,可以求得 Hopfield 的输出,这个输出反馈到输入从而产生新的输出,这个反馈过程一直进行下去。如果 Hopfield 网络是一个能收敛的稳定网络,则这个反馈与迭代的计算过程所产生的变化越来越小,一旦到达了稳定平衡状态;那么 Hopfield 网络就会输出一个稳定的恒值。对于一个 Hopfield 网络来说,关键在于确定它在稳定条件下的权系数。对于 Hopfield 网络来说,还存在如何判别它是稳定网络,还是不稳定网络的问题;而判别依据也是需要确定的。

Hopfield 最早提出的网络是二值神经网络,神经元的输出只取 -1 和 1 ,所以,也称离散 Hopfield 神经网络(Discrete Hopfield Neural Network, DHNN)。在离散 Hopfield 网络中,所采用的神经元是二值神经元,所输出的离散值 -1 和 1 分别表示神经元处于抑制和激活状态。

2. 离散 Hopfield 网络拓扑结构

如图 6-33 所示为是由 n 个神经元组成的离散 Hopfield 神经网络,第 1 层仅是作为网络的输入,它不是实际神经元,所以没有计算功能;而第 2 层是实际神经元,故而执行对输入信息与权系数相乘其积求累加和,并由非线性函数 f 处理后产生输出信息。 f 是一个简单的阈值函数,如果神经元的输出信息大于阈值 θ ,那么,神经元的输出就取值为 1 ;小于阈值 θ ,则神经元的输出就取值为 -1 。

对于一个离散的 Hopfield 网络,其网络状态是输出神经元信息的集合。对于一个输出层是 n 个神经元的网络,则其 t 时刻的状态为一个 n 维向量:

$$\mathbf{Y}(t) = [y_1(t), y_2(t), \dots, y_n(t)]$$

因为 $y_i(t)$ 取值为 $+1$ 或 -1 ,所以网络有 2^n 个状态,即是网络状态。

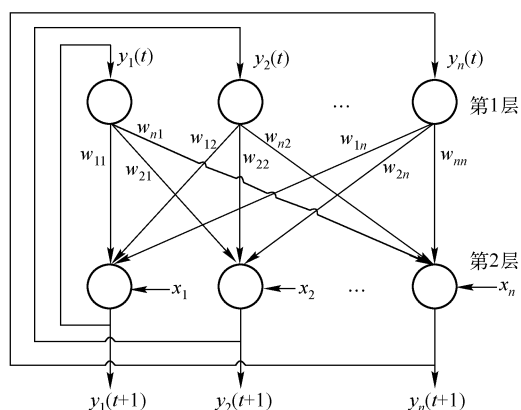


图 6-33 离散 Hopfield 网络拓扑结构图

3. 离散 Hopfield 网络结构的工作方式

(1) 同步(并行)方式

在时刻 t 时,所有神经元的状态都产生了变化,这时称为同步(并行)工作方式。

$$y_j(t+1) = f\left(\sum_{i=1}^n w_{ji}y_i(t) + x_j - \theta_j\right), j = 1, 2, \dots, n \quad (6-54)$$

不考虑外部输入时:

$$y_j(t+1) = f\left(\sum_{i=1}^n w_{ji}y_i(t) - \theta_j\right), j = 1, 2, \dots, n \quad (6-55)$$

(2) 异步(串行)方式

在时刻 t 时,只有某一个神经元 j 的状态产生变化,而其他 $n-1$ 个神经元的状态不变,这时称为异步工作方式。此时:

$$y_j(t+1) = f\left(\sum_{i=1}^n w_{ji}y_i(t) + x_j - \theta_j\right) \quad (6-56)$$

$$y_i(t+1) = y_i(t), i \neq j \quad (6-57)$$

不考虑外部输入时:

$$y_j(t+1) = f\left(\sum_{i=1}^n w_{ji}y_i(t) - \theta_j\right) \quad (6-58)$$

按照异步工作方式,某一时刻网络中只有一个结点被选择进行状态更新,当该结点状态变化时,网络状态就以其概率转移到另一状态;当该结点状态保持时,网络状态更新结果保持前一时刻的状态。通常,网络从某一初始状态开始经过多次更新后,才可能达到某一稳态。使用异步状态更新策略有若干好处:首先,算法实现容易,每个神经元结点有自己的状态更新时刻,不需要同步机制;其次,以异步方式更新网络的状态可以限制网络的输出状态,避免不同稳态以等概率出现。一旦给出 Hopfield 的权值和神经元的阈值,则网络的状态转移序列就确定了。

4. 离散 Hopfield 网络训练和分类识别方法

Hopfield 网络的训练和分类利用的是 Hopfield 网络的联想记忆功能,也称做联想存储器。

这是人类的智能特点之一。人类的所谓“触景生情”就是见到一些类同过去接触的景物,容易产生对过去情景的回味和思忆。由于网络可收敛于稳定状态,因此可用于联想记忆。若将稳态视为一个记忆,则由初始状态向稳态收敛的过程就是寻找记忆的过程,初态可视为是给定的部分信息,收敛过程可认为是从部分信息找到了全部信息,则实现了联想记忆的功能。联想记忆的一个重要的特性是由噪声输入模式反映出训练模式,这一点正是分类识别所需要的。

对于 Hopfield 网络,用它做联想记忆时,首先通过一个学习训练过程确定网络中的权系数,使所记忆的信息在网络的 n 维超立方体的某一个顶角的能量最小。当网络的权系数确定之后,只要向网络给出输入向量,这个向量可能是局部数据,即不完全或部分不正确的数据,但是网络仍然产生所记忆的信息的完整输出。1984 年 Hopfield 开发了一种用 n 维 Hopfield 网络做联想存储器的结构。在这个网络中,权系数的赋值规则为存储向量的外积存储规则(out product storage prescription)。

(1) 确定参数

① 确定输入向量 \mathbf{X} 。设有 N 个训练样品特征向量 $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \dots, \mathbf{X}_N$, 所有特征已经标准化。其特征空间为 n 维, 即 $\mathbf{X}_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T$, 将这 N 个训练样品存入 Hopfield 网络中, 则在网络中第 i, j 两个结点之间的权系数的值按式(6-59)计算, 完成网络的训练。

② 确定输出向量 \mathbf{Y} 。

输出向量 $\mathbf{Y} = [y_1, y_2, \dots, y_n]^T$ 。

③ 计算连接权值 w_{ji} 。连接权值 $w_{ji} (i = 1, 2, \dots, n; j = 1, 2, \dots, n)$ 。

$$w_{ji} = \begin{cases} \sum_{k=1}^N x_{ki} x_{kj}, & i \neq j \\ 0, & i = j \end{cases} \quad (6-59)$$

(2) 对待测样品分类

对于待测样品 $\mathbf{X} = [x_1, x_2, \dots, x_n]^T$, 通过对 Hopfield 网络构成的联想存储器进行联想检索过程实现分类功能。

① 将 \mathbf{X} 中各个分量的 x_1, x_1, \dots, x_n 分别作为第一层网络结点, n 个输入, 则结点有相应的初始状态 $\mathbf{Y}(t=0)$, 即 $y_j(0) = x_j, j = 1, 2, \dots, n$ 。

② 对于二值神经元, 计算当前 Hopfield 网络输出:

$$U_j(t+1) = \sum_{i=1}^n w_{ji} y_i(t) + x_j - \theta_j \quad j = 1, 2, \dots, n \quad (6-60)$$

$$y_j(t+1) = f(U_j(t+1)) \quad j = 1, 2, \dots, n \quad (6-61)$$

式中, x_j 为外部输入; f 是非线性函数, 可以选择阶跃函数; θ_j 为阈值参数。

$$f(U_j(t+1)) = \begin{cases} -1, & U_j(t+1) < 0 \\ +1, & U_j(t+1) \geq 0 \end{cases} \quad (6-62)$$

③ 对于一个网络来说, 稳定性是一个重大的性能指标。对于离散 Hopfield 网络, 其状态为 $\mathbf{Y}(t)$ 。如果, 对于任何 $\Delta t > 0$ 。当网络从 $t=0$ 开始, 有初始状态 $\mathbf{Y}(0)$; 经过有限时刻 t , 有 $\mathbf{Y}(t+\Delta t) = \mathbf{Y}(t)$, 则称网络是稳定的。此时的状态称为稳定状态。通过网络状态不断变化, 最后状态会稳定下来, 最终的状态是和待测样品向量 \mathbf{X} 最接近的训练样本向量。所以,

Hopfield 网络的最终输出也就是待测样品向量联想检索结果。

④ 利用最终输出与训练样品进行匹配,找出最相近的训练样本向量,其类别即是待测样品类别。这个过程说明,即使待测样品并不完全或部分不正确,也能找到正确的结果。在本质上,它具有滤波功能。

6.7.2 Hopfield 神经网络分类器设计

1. 实现步骤

① 初始化十个标准数字的训练样品,将其进行二值化处理。

② 调用 Matlab 的 newhop 函数,训练 Hopfield 网络。

newhop 函数定义为: $\text{net} = \text{newhop}(T)$,其中 T 为训练样品集合,net 为训练后的 Hopfield 网络结构名。

③ 对于一个未知类别的待测样品,调用 sim 函数,利用已训练好的 Hopfield 网络进行仿真计算,最终网络的稳定状态下的输出保存到向量 t 中。

④ 将网络输出向量 t 与标准数字训练样品进行匹配,找出最接近的训练样品,其类别号即为待测样品类别,从而实现了分类识别。

2. 编程代码

```
%%%%%%%%%%%%%%
% 函数名称:Hopfield
% 函数功能:使用 Hopfield 网络,识别手写数字
% 函数参数:手写数字特征
% 函数返回值:手写数字所属类别 y
%%%%%%%%%%%%%
function y = Hopfield( samplehop );
    clc;
    zero = [ -1 1 1 1 -1 1 1 -1 1 1 1 -1 -1 -1 1 1 -1 1 1 -1 ];
    one = [ -1 -1 1 -1 -1 -1 -1 1 -1 -1 -1 -1 1 -1 -1 -1 -1 -1 -1 ];
    two = [ 1 1 1 1 1 -1 -1 -1 -1 1 -1 -1 1 1 1 -1 -1 -1 1 1 ];
    three = [ 1 1 1 1 1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1 1 1 ];
    four = [ -1 -1 1 1 -1 -1 1 1 1 -1 1 1 -1 1 1 1 1 -1 -1 -1 ];
    five = [ -1 -1 1 1 1 -1 1 -1 -1 -1 1 1 1 -1 -1 -1 -1 1 1 1 ];
    six = [ -1 -1 1 -1 -1 -1 1 -1 -1 -1 1 1 1 1 1 -1 -1 -1 1 1 ];
    seven = [ 1 1 1 1 1 -1 -1 -1 1 -1 -1 -1 1 -1 -1 -1 -1 1 1 ];
    eight = [ 1 1 1 1 1 1 1 -1 1 -1 -1 -1 1 -1 1 1 1 1 1 1 ];
    nine = [ 1 1 1 1 1 1 -1 -1 -1 1 1 1 1 -1 -1 1 -1 -1 -1 ];
    T = [ zero;one;two;three;four;five;six;seven;eight;nine ]';
    net = newhop( T );
    [ Y,Pf,Af ] = sim( net, { 1 5 }, { }, { samplehop } );
    for i = 1:25
        if( Y{1}( i,1 ) > 0.5 )
```

```

        Y{1}(i,1) = 1;
    else
        Y{1}(i,1) = -1;
    end
end
a = [0 0 0 0 0 0 0 0 0];
b = -1;
temy = 0;
for i = 1:10
    for j = 1:25
        if Y{1}(j,1) ~= T(j,i)
            continue;
        end
        if Y{1}(j,1) == T(j,i)
            a(1,i) = a(1,i) + 1;
        end
    end
    if(a(1,i) > b)
        b = a(1,i);
        temy = i-1;
    end
end
y = temy

```

3. 效果图

拖动鼠标左键在视图区用鼠标手写一个数字,如图 6-34 所示,然后选择“神经网络”→“Hopfield 神经网络分类法”→“Hopfield 神经网络识别”菜单命令,进行手写数字分类。Hopfield 手写数字的识别效果,如图 6-35 所示。

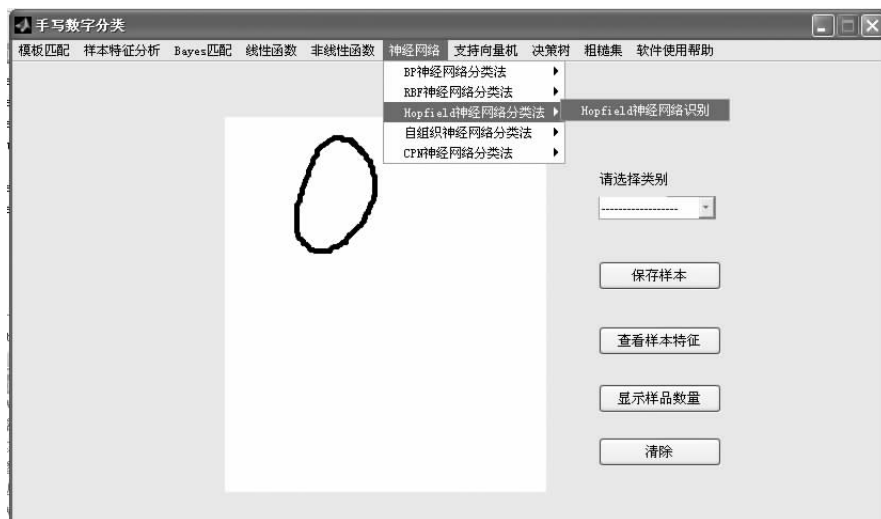


图 6-34 Hopfield 神经网络识别

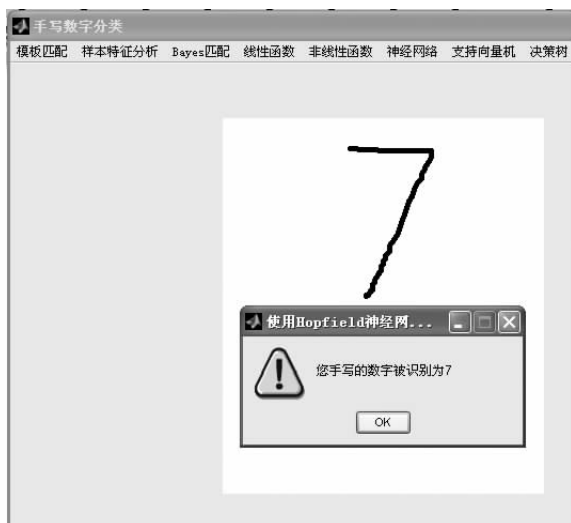


图 6-35 Hopfield 手写数字的识别效果

本章小结

近年来,神经网络已成为研究的热点,并取得了广泛的应用。人工神经网络是从输入空间到输出空间的一个非线性映射,通过调整权重和阈值来“学习”或发现变量间的关系,实现对事务的分类。本章介绍了人工神经网络的基本原理,包括人工神经元、人工神经网络模型、神经网络的学习过程、人工神经网络在模式识别问题上的优势,并应用 BP 神经网络、径向基函数神经网络、自组织竞争神经网络、概率神经网络、对向传播神经网络和反馈型神经网络实现模式识别的基本原理和具体方法,并给出了 Matlab 可执行程序,对理论进行了验证。实践证明,神经网络是一种对数据分布无任何要求的非线性技术,它能有效解决非正态分布、非线性的评价问题,因而受到广泛的应用。

习题 6

1. 简述人工神经网络在模式识别问题上的优势。
2. 试述 BP 神经网络的拓扑结构及其学习算法。
3. 简述设计 BP 神经网络需要考虑的主要因素。
4. 试述基于高斯核的 RBF 神经网络的拓扑结构及其初始权值的产生方式。
5. 简述概率神经网络(PNN)的拓扑结构及其各层的功能。
6. 简述 Hopfield 神经网络的拓扑结构和稳定条件。
7. 简述离散型 Hopfield 神经网络的学习算法,并试用 C 语言实现离散型 Hopfield 神经网络。
8. 简述对向传播神经网络(CPN)的拓扑结构、基本思想及其学习算法。
9. 简述自组织神经网络的拓扑结构、基本思想及其训练过程。
10. 试列表对比本章介绍的几种神经网络的异同。

第 7 章 决策树分类器设计

本章要点:

- ☑ 决策树的基本概念
- ☑ 决策树分类器设计

7.1 决策树的基本概念

1. 决策树的基本原理

决策树 (Decision Tree) 又称判定树, 是用于分类和预测的一种树结构。决策树学习是以实例为基础的归纳学习算法。它着眼于从一组无次序、无规则的实例中推理出决策树表示形式的分类规则。它采用自顶向下的递归方式, 在决策树的内部结点进行属性值的比较并根据不同属性判断从该结点向下的分支, 在决策树的叶结点得到结论。所以从根结点开始对应着一条合取规则, 整棵树就对应一组析取表达式规则。

决策树中的每个内部结点 (Internal Node) 代表对某一属性的一次测试, 每条边代表一个测试结果, 叶结点 (Leaf) 代表某个类 (Class) 或类的分布 (Class Distribution)。

下例是一棵决策树, 从中可以看到决策树的基本组成部分: 决策结点、分支和叶结点。

[例 1] 如图 7-1 所示, 给出了一个决策树的例子, 从中可以看出一位用户是否会买汽车, 用它可以预测某个人的购买意向。

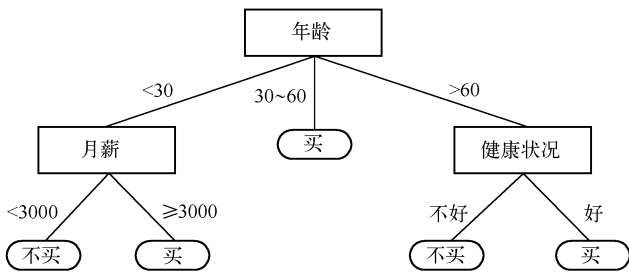


图 7-1 买车问题的决策树

这棵决策树对销售记录进行分类, 指出一个消费者是否会购买汽车。每个内部结点 (方形框) 代表对某个属性的一次检测。每个叶结点 (椭圆框) 代表一个类: 买车或不买车。

在这个例子中, 样品向量为: (年龄, 月薪, 健康状况; 买车意向), 待测样品格式为: (年龄, 月薪, 健康状况), 输入新的待测样品记录, 可以预测该待测样品隶属于哪个类。

构造决策树采用自上而下的递归构造方法。以多叉树为例, 构造思路是: 如果训练集中所有数据是同类的, 则将之作为叶子结点, 结点内容即是该类别标记, 否则根据某种策略选择一个属性, 按照属性的各个取值, 把数据集合划分若干个子集, 使得每个子集上的所有数据在该

属性上具有同样的属性值;然后再依次递归处理各个子集。这种思路称之为“分而治之”。

决策树构造的结果是一棵二叉或多叉树,它的输入是一组带有类别标记的训练数据。二叉树的内部结点(非叶结点)一般表示为一个逻辑判断,如形式为 $(a=b)$ 的逻辑判断,其中 a 是属性, b 是该属性的某个属性值;树的边是逻辑判断的分支结果。多叉树(ID3)的内部结点是属性,边是该属性的所有取值,有几个属性值,就有几条边。树的叶结点都是类别标记。

2. 决策树分类

决策树分类算法起源于概念学习系统 CLS(Concept Learning System),然后发展到 ID3 方法达到高潮,最后又演化为能处理连续属性的 C4.5,此外,决策树方法还有 CART、SLIQ、SPRINT 等。最初的算法利用信息论中信息增益方法寻找训练集中具有最大信息量的字段,把决策树的一个结点字段的某些值作为分水岭建立树的分支;在分支下建立下层结点和子分支,生成一棵决策树。再剪枝,优化,然后把决策树转化为规则,利用这些规则可以对新事例进行分类。

使用决策树进行分类分为两步:

步骤1 建立决策树模型:利用训练集建立并精化一棵决策树。这个过程实际上是一个从数据中获取知识,进行机器学习的过程。这个过程通常分为两个阶段:

① 建树(Tree Building):这是一个递归的过程,最终将得到一棵树。

② 剪枝(Tree Pruning):剪枝的目的是降低由于训练集存在噪声而造成的起伏。

步骤2 利用生成完毕的决策树对输入数据进行分类:对输入的待测样品,从根结点依次测试记录待测样品的属性值,直到到达某个叶结点,从而找到该待测样品所在的类。

3. 决策树方法的特点

与其他分类方法相比,决策树分类有如下优点:

① 分类速度快,计算量相对较小,容易转化成分类规则。只须沿着树根向下一直到叶结点,沿途的分裂条件就能够唯一确定一条分类的谓词。如在例1中,“年龄→健康状况→不买”这条路径谓词表示为“如果一个人年龄大于60且身体不好,那么他不会买车”。

② 分类准确性高,从决策树中挖掘出的规则准确性高且便于理解。

当然,一般决策树方法也存在缺乏伸缩性,处理大训练集时算法的额外开销大,降低了分类的准确性。

7.2 决策树分类器设计

1. 理论基础

Quinlan 提出的 ID3 算法是决策树算法的代表,具有描述简单、分类速度快的优点,大多数决策树算法都是在它的基础上加以改进而实现的。它采用分治策略,通过选择窗口形成决策树,利用信息增益寻找训练集数据库中具有最大信息量的属性,建立决策树的一个结点,再根据该属性的不同取值建立树的分支,在每个分支子集中重复建立树的下层结点和分支过程。

(1) ID3 算法的基本思想

① 任意选取一个属性作为决策树的根结点,然后就这个属性所有的取值创建树的分支。

② 用这棵树来对训练集进行分类,如果一个叶结点无标记且该结点的所有实例都属于同一类,则以该类为标记标识此叶结点;如果所有的叶结点都有类标记,则算法终止。

③ 否则,选取一个从该结点到根路径中没有出现过的属性为标记,标识该结点,然后就这个属性所有的取值继续创建树的分支;重复算法步骤②。

这个算法一定可以创建一棵基于训练集的正确决策树,然而,这棵决策树不一定是最简单的。显然,不同的属性选取顺序将生成不同的决策树。因此,适当地选取属性将生成一棵简单的决策树。在 ID3 算法中,采用了一种基于信息的启发式的方法来决定如何选取属性。启发式方法选取具有最高信息量的属性,也就是说,生成最少分支决策树的那个属性。

(2) 属性选择度量

ID3 算法在树的每个结点上以信息增益 (Information gain) 作为度量来选择测试属性。这种度量称为属性选择度量或分裂的优良性度量。选择具有最高信息增益 (或最大熵压缩) 的属性作为当前结点的测试属性。该属性使得对结果划分中的样本分类所需要的信息量最小,并确保找到一棵简单的 (但不一定是最简单的) 决策树。

Information Gain 的原理取自 1948 年香农 (C. E. Shannon) 提出的信息论。其中给出了关于信息量 (Information) 和熵 (Entropy) 的定义,熵实际上是系统信息量的加权平均,也就是系统的平均信息量。

定义 1 期望信息量: 设训练集为 \tilde{X} , 样品总数为 N , 其中包含 M 个不同的类 $\omega_i (i = 1, 2, \dots, M)$ 。设 N_i 是 \tilde{X} 中属于类 ω_i 的样品的个数。对一个给定样品分类所需的期望信息为

$$I(N_1, N_2, \dots, N_M) = - \sum_{i=1}^M p_i \log_2(p_i) \quad (7-1)$$

其中 p_i 是样品属于 ω_i 的概率, 用 N_i/N 来估计。

定义 2 熵: 属性 A 是有 k 个不同值的属性 $\{a_1, a_2, \dots, a_j, \dots, a_k\}$, A 可以把全体训练集 \tilde{X} 分成 k 个子集 S_1, S_2, \dots, S_k , 其中 $S_j = \{X | X \in \tilde{X}, X.A = a_j\}$ 。如果 A 选为测试属性, 那么那些子集表示从代表集合 \tilde{X} 出发的所有树枝。设 N_{ij} 表示 S_j 中类为 ω_i 的样品的个数。根据属性 A 划分的子集的熵, 也就是系统总熵为

$$E(A) = \sum_{j=1}^k \left[\left(\frac{N_{1j} + N_{2j} + \dots + N_{Mj}}{N} \right) \cdot I(N_{1j}, N_{2j}, \dots, N_{Mj}) \right] \quad (7-2)$$

式中, $\left(\frac{N_{1j} + N_{2j} + \dots + N_{Mj}}{N} \right)$ 表示第 j 个子集的权重; N 为训练集 \tilde{X} 中样品个数。对于给定子集, S_j

$$I(N_{1j}, N_{2j}, \dots, N_{Mj}) = - \sum_{i=1}^M p_{ij} \log_2(p_{ij}) \quad (7-3)$$

式中, $p_{ij} = N_{ij}/|S_j|$ 表示 S_j 中的样本属于 ω_i 的概率; $|S_j|$ 表示 S_j 中的样品个数。

定义 3 在属性 A 上分支获得的信息增益表示为

$$\text{Gain}(A) = I(N_1, N_2, \dots, N_M) - E(A) \quad (7-4)$$

$\text{Gain}(A)$ 是指由于知道属性 A 的值而导致的熵的期望压缩。熵是一个衡量系统混乱程度的统计量。熵越大, 表示系统越混乱。分类的目的是提取系统信息, 使系统向更加有序、有规则、有组织的方向发展。所以最佳的分裂方案是使熵减少量最大的分裂方案。熵减少量就是 Information Gain, 所以, 最佳分裂就是使 $\text{Gain}(A)$ 最大的分裂方案。通常, 这个最佳方案是用

“贪心算法+深度优先搜索”得到的。算法计算每个属性的信息增益。具有最高信息增益的属性选做给定集合 S 的测试属性,创建一个结点,并以该属性标记,对属性的每个值创建分支,据此划分样本。

【例2】如表7-1所示的是一个顾客买车意向的训练集,通过此例来说明属性选择方法。

表7-1 顾客买车意向的训练集

样品编号	年龄	月薪	健康状况	买车意向(类别)
1	<30	<3000	好	不买
2	<30	<3000	不好	不买
3	<30	≥ 3000	不好	买
4	<30	≥ 3000	好	买
5	30~60	<3000	好	买
6	30~60	≥ 3000	好	买
7	30~60	≥ 3000	不好	买
8	>60	<3000	好	买
9	>60	<3000	不好	不买
10	>60	≥ 3000	不好	不买

从表7-1中可以看出,类属性“买车意向”有两个不同的值{买,不买},因此一共有两个类 $M=2$ 。设 ω_1 对应于“买”, ω_2 对应于“不买”,则 ω_1 有6个样本, $N_1=6$, ω_2 有4个样本, $N_2=4$ 。首先利用式(7-1)计算期望信息 $I(N_1, N_2)$ 。

$$I(N_1, N_2) = I(6, 4) = -\frac{6}{10} \log_2 \frac{6}{10} - \frac{4}{10} \log_2 \frac{4}{10} = 0.9710$$

然后计算每个属性的熵。对于属性“年龄”,有三种取值,即三个子集,分别计算三个子集的期望信息。

$$\text{年龄} = "<30": N_{11} = 2, N_{21} = 2, I(N_{11}, N_{21}) = -\frac{2}{4} \log_2 \frac{2}{4} - \frac{2}{4} \log_2 \frac{2}{4} = 1$$

$$\text{年龄} = "30 \sim 60": N_{12} = 3, N_{22} = 0, I(N_{12}, N_{22}) = -\frac{3}{3} \log_2 \frac{3}{3} = 0$$

$$\text{年龄} = ">60": N_{13} = 1, N_{23} = 2, I(N_{13}, N_{23}) = -\frac{1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} = 0.9183$$

根据式(7-2),计算样本按“年龄”划分成子集的熵为

$$E(\text{年龄}) = \frac{4}{10} I(N_{11}, N_{21}) + \frac{3}{10} I(N_{12}, N_{22}) + \frac{3}{10} I(N_{13}, N_{23}) = 0.6755$$

信息增益为

$$\text{Gain}(\text{年龄}) = I(N_1, N_2) - E(\text{年龄}) = 0.2955$$

同理,我们可以得到其余两个属性的信息增益:

$$\text{Gain}(\text{月薪}) = 0.1246$$

$$\text{Gain}(\text{健康状况}) = 0.1246$$

由于“年龄”属性具有最高信息增益,因此被选择为测试属性。创建一个以年龄为标记的

结点,对每一属性值引出一个分支,如图 7-2 所示。

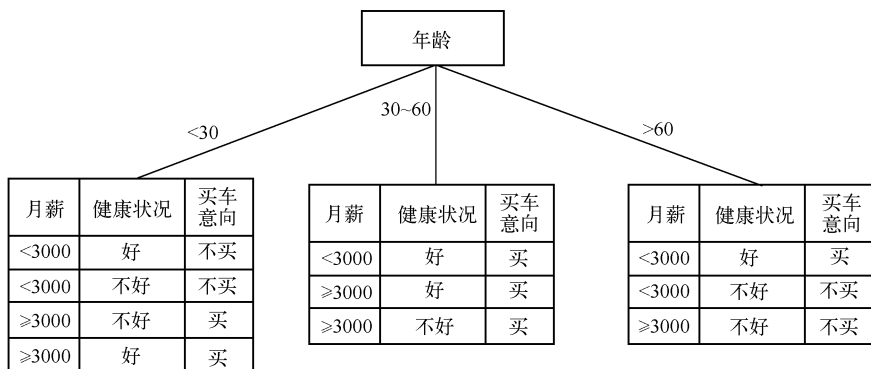


图 7-2 对属性“年龄”进行划分

从图 7-2 中可以看出,落在分支“30 ~ 60”的样本都属于同一类,因此该分支结点为一个叶结点。然后对另外两个结点子集继续进行属性选择,创建分支,直至分支结点全部为叶结点,生成图 7-1 的决策树。

(3) 决策树剪枝

由于训练集中的数据一般不可能是完美的,有些属性缺值或不准确,即存在噪声数据。基本的决策树构造算法没有考虑噪声和孤立点,生成的决策树完全与训练集拟合。有噪声情况下,完全拟合将导致过分适应。由于数据中的噪声和孤立点,许多分支反映的是训练集中的异常。剪枝阶段的任务就是利用统计学方法,去掉最不可靠、可能是噪声的一些枝条对决策树的影响即过分适应数据问题,从而提高独立于测试数据正确分类的能力来达到净化树的目的。当然,当数据稀疏时,要防止过分剪枝。因此,剪枝对有些数据效果好而有些数据则效果差。

剪枝方法主要有两类:先剪枝和后剪枝。

① 先剪枝(pre-pruning)。在建树的过程中,当满足一定条件,例如,Information Gain 或者某些有效统计量达到某个预先设定的阈值时,结点不再继续分裂,内部结点成为一个叶结点。叶结点取子集中频率最大的类作为自己的标识,或者可能仅仅存储这些实例的概率分布函数。

② 后剪枝(pos-pruning)。它是由“完全生长”的树剪去分支,首先生成与训练数据集合完全拟合的一棵决策树,然后从树的叶子开始剪枝,逐步向根的方向剪。剪枝时要用到一个测试数据集合(Adjusting set),如果存在某个叶子剪去后能使得在测试集上的准确度不降低,则剪去该叶子,最终形成一棵错误率尽可能小的决策树。

如果在训练集中出现了类交叉的情况,也就是说,在待挖掘的数据中出现矛盾和不一致,将出现这样一种情况:在一个树结点中,所有的实例并不属于一个类却找不到可以继续分支的属性。

ID3 算法使用以下两种方案解决这个问题:

- 选择在该结点中所占比例最大的类为标记并标识该结点;
- 根据该结点中不同类的概率分布为标记并标识该结点。

如果在训练集中出现了某些错误的实例,即在待挖掘的数据中,本来应该属于同一结点的数据因为某些错误的属性取值而继续分支,则在最终生成的决策树中可能出现分支过细和错误分类的现象。ID3 算法设置了一个阈值来解决这个问题:只有属性的信息量超过这个阈值时才创

建分支,否则以类标志标识该结点。该阈值的选取对决策树的正确创建具有相当重要的作用。如果阈值过小,可能没有发挥应有的作用;如果阈值过大,又可能删除了应该创建的分支。

(4) 从决策树提取分类规则

从决策树中可以提取分类规则,并以 IF-THEN 的形式表示。具体方法是:从根结点到叶结点的每一条路径创建一条分类规则,路径上的每一对“属性-值”对应规则的前件(即 IF 部分)的一个合取项,叶结点为规则的后件(即 THEN 部分)。

对于例 1 中的决策树可提取以下分类规则:

IF 年龄 = ‘<30’ AND 月薪 = ‘<3000’ THEN 买车意向 = ‘不买’。

IF 年龄 = ‘<30’ AND 月薪 = ‘≥3000’ THEN 买车意向 = ‘买’。

IF 年龄 = ‘30~60’ THEN 买车意向 = ‘买’。

IF 年龄 = ‘>60’ AND 健康状况 = ‘不好’ THEN 买车意向 = ‘不买’。

IF 年龄 = ‘>60’ AND 健康状况 = ‘好’ THEN 买车意向 = ‘买’。

(5) ID3 算法的改进

基本的 ID3 算法采用信息增益作为属性的度量,试图减少树的平均深度,而忽略了对叶子数目的研究,导致了许多问题:最优属性选取不准确,信息增益的计算依赖于属性取值数目较多的特征,而取值较多的属性不一定是最优属性;抗噪性差,训练集中正例和反例较难控制。因此,众多学者针对 ID3 算法的不足,提出了许多改进策略。

① 离散化。对符号性属性(即离散性属性)ID3 算法的知识挖掘比较简单,算法将针对属性的所有符号创建决策树分支。但是,如果属性值是连续的,如本书实例中手写数字中的特征值等,针对属性的所有不同的值创建决策树,则将由于决策树过于庞大而导致算法失效。

为了解决该问题,在用 ID3 算法挖掘具有连续性属性的知识时,应该首先把该连续性属性离散化。最简单的方法就是把属性值分成两段。如手写数字中的特征值可以分为大于 0.3 和小于 0.3 等。因此选择最佳的分段值是离散化的核心。对任何一个属性,其所有的取值在一个数据集中是有限的。假设该属性取值为 $\{a_1, a_2, \dots, a_n\}$,则在这个集合中,一共存在 $n-1$ 个分段值, ID3 算法采用计算信息量的方法计算最佳的分段值,然后进一步构建决策树。

② 空缺值处理。训练集中的数据可能会出现某一训练样本中某一属性值空缺的情况,因此必须进行空缺值处理。可以采用如下的空缺值处理方法:若属性 A 有空缺值,则可用 A 的最常见值、平均值、样本平均值等填充。

③ 属性选择度量。在决策树建树过程中,有许多的属性选择度量方法,对于一个特定的数据集来讲,只有“适合”与“不适合”之分,而没有“好”与“差”之分,与数据集中属性的值的多少,类别的多少有关系。ID3 算法中采用信息增益作为属性选择度量,但它仅适合于具有多个值的属性。还有一些其他的属性选择度量方法,如增益率,可伸缩性指标基尼指数等。

④ 可伸缩性。ID3 算法对于相对较小的训练数据集是有效的,但对于现实世界中数以百万计的训练数据集,需要频繁地将训练数据在主存和高速缓存中换进换出,从而使算法的性能低下。因此可以将训练数据集划分为几个子集,使得每个子集能够放入内存;然后由每个子集构造一棵决策树;最后,将每个子集得到的分类规则组合起来,得到输出的分类规则。

⑤ 碎片、重复和复制处理。碎片是指在一个给定的分支中的样本数太少,从而失去统计意义。解决的方法是将分类属性值分组,决策树结点可以测试一个属性值是否属于给定的集

2. 实现步骤

(1) 构建训练样本集

(2) 构建分类决策树

(3) 利用决策树分类

(4) 显示决策树

显示决策树函数为 `treedisp(T)`。参数 T 为已构建的决策树结构。

3. 编程代码

```
% 函数名称:DecisionTree( )
% 参数:sample:待测样品
% 返回值:result:分类结果
% 函数功能:决策树分类算法
function result = DecisionTree( sample );

x = [ ];
y = cell(1,1000);
for i = 1:10
    % 初始化训练样本集
    x = [ x pattern(i).feature(:,1:100) ];
    str = num2str(i-1);
```

```

% 初始化类别
y(1,100 * (i - 1) + 1:100 * i) = {str};
end
% 构造分类决策树
t = treefit(x',y);
% 决策树分类
result = treeval(t,sample);
sfit = t.classname(result);
% 获得分类结果
result = str2num(sfit{1,1});
% 显示分类结果
treedisp(t);

```

4. 效果图

首先手写一个数字,如图 7-3(a)所示,单击“决策树分类”按钮,进行决策树算法分类,得到正确的分类结果,如图 7-3(b)所示。同时,显示决策树结构图。如图 7-3(c)所示为决策树结构的一部分,图中空心三角形表示分支结点,分支结点旁边的不等式表示分支规则,例如,图中根结点含义是如果第 22 个特征小于 0.0205779 则进入左边分支,否则进入右边分支。实体圆形表示叶结点,叶结点下方的数字为分类类别。

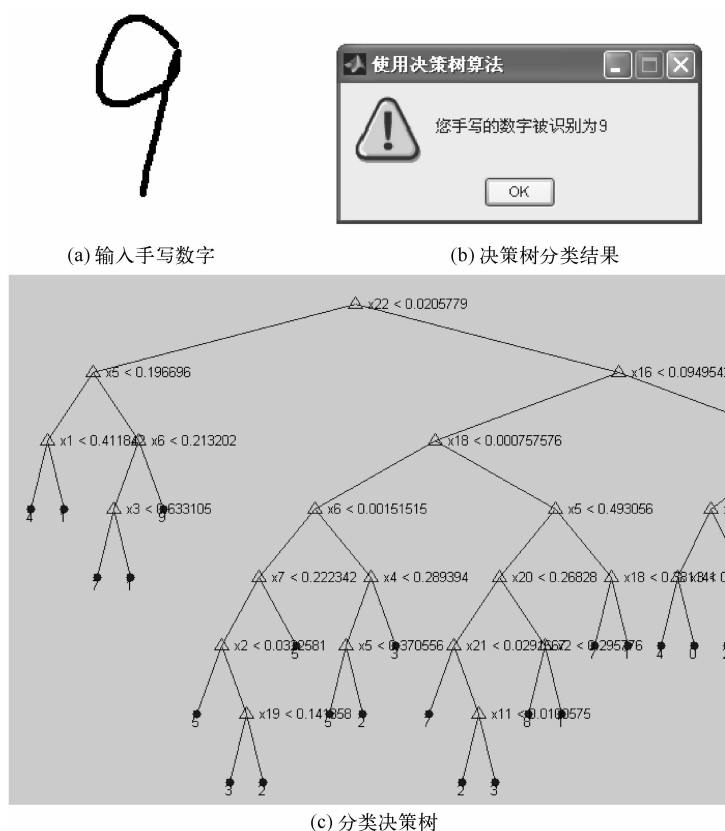


图 7-3 决策树分类算法效果图

本章小结

本章介绍了决策树理论的基本概念,包括决策树的基本原理、决策树分类方法、决策树方法的特点等;详细介绍了基于决策树理论的分类方法,包括 ID3 算法基本思想,属性选择度量等;最后介绍了基于决策树理论的手写数字分类器设计实现。

习题 7

- 1. 简述决策树的构造方法。
- 2. 简述 ID3 算法基本思想以及基于信息增益的属性选择度量方法。
- 3. 对于一个训练集,如表 7-2 所示,利用 ID3 算法构造一棵决策树。

表 7-2 流感决策问题训练集

样 品 编 号	咳 嗽	头 晕	体 温	流感(类别)
1	是	是	正常	否
2	是	是	高	是
3	是	是	非常高	是
4	否	是	正常	否
5	否	否	高	否
6	否	是	非常高	是
7	是	否	高	是
8	否	是	正常	否

- 4. 叙述决策树理论在分类问题中的实现方法和步骤。

第 8 章 粗糙集分类器设计

本章要点:

- ☑ 粗糙集理论的基本概念
- ☑ 粗糙集理论在模式识别中的应用
- ☑ 粗糙集分类器设计

数据是对客观事物的属性、数量、位置或它们之间的相互关系的形式表示,是各种信息的载体。但是自然界中大部分事物所呈现的信息都是不完整和模糊的。在经典逻辑中,只有真假二值之分,因而无法对此类问题进行准确的描述。长期以来许多逻辑学家和哲学家都致力于研究模糊概念。在现实世界中,有许多模糊现象不能简单地用好坏、真假来表示。如何较好地表示和处理这些现象就成为一个问题。问题集中在分类边界上,也就是说,存在一些个体,即不能说它属于某个子集,也不能说它不属于某个子集。

粗糙集(Rough set)是波兰数学家 Z. Pawlak 于 1982 年提出的。粗糙集以等价关系为基础,用于分类问题。它用上、下近似两个集合来逼近任意一个集合,该集合的边界区域被定义为上近似集和下近似集之差集,边界区域就是那些无法归属的个体。上、下近似两个集合可以通过等价关系给出确定的描述,边界域的元素数目可以被计算出来。

从本质上看,粗糙集理论反映了认知过程在非确定、非模型信息处理方面的机制和特点,是一种有效的非单调推理工具。该理论在数据的决策和分析、模式识别、机器学习与知识发展等方面有着成功的应用,已成为信息科学最活跃的研究领域之一。目前,该理论还在医学、化学、材料学、地理学、管理科学和金融等其他学科得到了成功的应用。

8.1 粗糙集理论的基本概念

1. 知识表达系统和决策表

知识是对某些客观对象的认识。为了处理智能数据,需要对知识进行符号表示。知识表达系统研究将对象的知识通过指定对象的基本特征和特征值来描述,以便通过一定的方法从大量的数据中发现有用的知识或决策规则。

知识表达系统可用一个四元组 S 表示:

$$S = (\tilde{X}, R, V, f)$$

式中, \tilde{X} 为一个非空有限对象的集合,称为论域 $\tilde{X} = \{X_1, X_2, \dots, X_N\}$, 其中 X_i 为对象;

R 为对象的属性集合;

V 为属性值的集合, V_a 是属性 $a \in R$ 的值域;

f 为 $\tilde{X} \times R \rightarrow V$ 的一个信息函数,它为每个对象 X 的每个属性 a 赋予一个属性值,即 $a \in R$, $X \in \tilde{X}, f_a(X) \in V_a$ 。

知识表达系统也称为信息系统,通常用 $S = (\tilde{X}, R)$ 代替 $S = (\tilde{X}, R, V, f)$ 。知识表达系统的数据以关系表的形式表示,关系表的行对应要研究的对象,列对应对象的属性,对象的信息通过指定对象的各属性值来表示。

设 $S = (\tilde{X}, R, V, f)$ 是一个信息系统(知识表达系统), $R = C \cup D$, C 称为条件属性集合, D 称为决策属性集。具有条件属性和决策属性的信息系统称为决策表。

对象的特征由条件属性描述,决策属性表示该对象的分类。决策属性可能表示专家根据条件属性所描述的情形所做的分类、采取的行动或决策。在机器学习意义上说,决策表被用做学习一个由条件属性产生的描述到由决策属性确定的分类的映射。通常,这个映射表示为一个决策规则(分类器或决策算法)的集合。

2. 等价关系

设 A 代表某种属性集合。 a 代表属性中的某一种取值。如果有两个样品 X_i, X_j 满足如下关系

对于 $\forall a \in A, A \subset R, X_i \in \tilde{X}, X_j \in \tilde{X}$, 它们的属性值相同,即 $f_a(X_i) = f_a(X_j)$ 成立,称对象 X_i 和 X_j 是对属性 A 的等价关系,表示为

$$IND(A) = \{(X_i, X_j) | (X_i, X_j) \in \tilde{X} \times \tilde{X}, \forall a \in A, f_a(X_i) = f_a(X_j)\}$$

可见属性值相同的两个样品之间的关系为等价关系。

3. 等价集

在 \tilde{X} 中,对属性集 A 中具有相同等价关系的元素集合成为等价关系 $IND(A)$ 的等价集, $[X]_A$ 表示在属性 A 下与 X 具有等价关系的元素集合。

$$[X]_A = \{X_j | (X, X_j) \in IND(A)\}$$

4. 等价划分

从所采集的训练集中把属性值相同的样品聚类,形成若干个等价集,构成 A 集合。在 \tilde{X} 中对属性 A 的所有等价集形成的划分表示为

$$A = \{E_i | E_i = [X]_A, i = 1, 2, \dots\}$$

具有特性:

- ① $E_i \neq \emptyset$;
- ② 当 $i \neq j$ 时, $E_i \cap E_j = \emptyset$;
- ③ $\tilde{X} = \cup E_i$ 。

[例1] 设 $\tilde{X} = \{a(\text{血压正常}), b(\text{血压正常}), c(\text{血压正常}), d(\text{血压高}), e(\text{血压高}), f(\text{血压很高})\}$, 对于属性 $A(\text{血压})$ 的等价关系有

$$IND(A) = \{(a, b), (a, c), (b, c), (d, e), (a, a), (b, b), (c, c), (d, d), (e, e), (f, f)\}$$

属性 A 的等价集有

$$E_1 = [a]_A = [b]_A = [c]_A = \{a, b, c\}$$

$$E_2 = [d]_A = [e]_A = \{d, e\}$$

$$E_3 = [f]_A = \{f\}$$

\tilde{X} 中对属性 A 的划分为

$$A = \{E_1, E_2, E_3\} = \{\{a, b, c\}, \{d, e\}, \{f\}\}$$

5. 上近似集和下近似集

属性 A 可划分为若干个等价集,与决策集 Y 对应关系分上近似集 $A^-(Y)$ 和下近似集 $A_-(Y)$ 两种:

(1) 下近似定义

对任意一个决策属性的等价集 $Y(Y \subseteq \tilde{X})$,属性 A 的等价集 $E_i = [X]_A$,有

$$A_-(Y) = \cup \{E_i | E_i \in A \wedge E_i \subseteq Y\}$$

或

$$A_-(Y) = \{X | [X]_A \subseteq Y\}$$

表示等价集 $E_i = [X]_A$ 中的元素 X 都属于 Y ,即 $\forall X \in A_-(Y)$,则 X 一定属于 Y 。 $A_-(Y)$ 表示下近似集。

(2) 上近似定义

对任意一个决策属性的等价集 $Y(Y \subseteq \tilde{X})$,属性 A 的等价集 $E_i[X]_A$,有

$$A^-(Y) = \cup \{E_i | E_i \in A \wedge E_i \cap Y \neq \emptyset\}$$

或

$$A^-(Y) = \{X | [X]_A \cap Y \neq \emptyset\}$$

表示等价集 $E_i = [X]_A$ 中的元素 X 可能属于 Y ,即 $\forall X \in A^-(Y)$,则 X 可能属于 Y ,也可能不属于 Y 。 $A^-(Y)$ 表示上近似。

(3) 正域、负域和边界的定义

全集 \tilde{X} 可以划分为 3 个不相交的区域,即正域(POS_A)、负域(NEG_A)和边界(BND_A):

正域: $POS_A(Y) = A_-(Y)$

负域: $NEG_A(Y) = \tilde{X} - A^-(Y)$

边界: $BND_A(Y) = A^-(Y) - A_-(Y)$

由此可见:

$$A^-(Y) = A_-(Y) + BND_A(Y)$$

如图 8-1 所示,形象地表示出正域、负域和边界,每一个小长方形表示一个等价集。

从图 8-1 中可以看出:任意一个元素 $X \in POS(Y)$,一定属于 Y ;任意一个元素 $X \in NEG(Y)$,一定不属于 Y ;集合 Y 的上近似是其正域和边界的并集,即

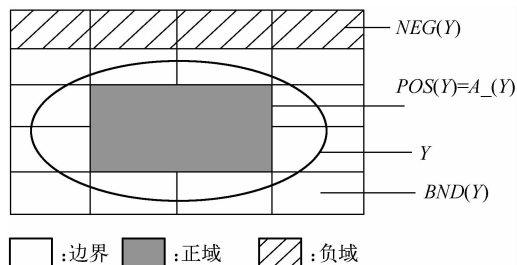


图 8-1 Y 关于属性 A 的正域、负域和边界

$$A^-(Y) = POS_A(Y) \cup BND_A(Y)$$

对于元素 $X \in BND(Y)$, 无法确定其是否属于 Y , 因此对于任意元素 $X \in A^-(Y)$, 只知道 X 可能属于 Y 。

6. 粗糙集

若 $A^-(Y) = A_-(Y)$, 即 $BND_A(Y) = \emptyset$, 即边界为空, 称 Y 为 A 的可定义集; 否则 Y 为 A 的不可定义集, 即 $A^-(Y) \neq A_-(Y)$, 称 Y 为 A 的粗糙集(Rough set)。

7. 确信度

确信度 $\alpha_A(Y)$ 表示为

$$\alpha_A(Y) = \frac{|\tilde{X}| - |A^-(Y) - A_-(Y)|}{|\tilde{X}|}$$

其中 $|\tilde{X}|$ 和 $|A^-(Y) - A_-(Y)|$ 分别表示集合 \tilde{X} 、 $A^-(Y) - A_-(Y)$ 中的元素个数。

$\alpha_A(Y)$ 的值反映了 \tilde{X} 中的能够根据 A 中各属性的属性值确定其属于或不属于 Y 的比例, 也即对 \tilde{X} 中的任意一个对象, 根据 A 中各属性的属性值确定它属于或不属于 Y 的可信度。

确信度性质:

$$0 \leq \alpha_A(Y) \leq 1$$

① 当 $\alpha_A(Y) = 1$ 时, \tilde{X} 中的全部对象能够根据 A 中各属性的属性值就可以确定其是否属于 Y , Y 为 A 的可定义集。

② 当 $0 < \alpha_A(Y) < 1$ 时, \tilde{X} 中的部分对象根据 A 中各属性的属性值可以确定其是否属于 Y , 而另一部分对象则不能确定其是否属于 Y , Y 为 A 的部分可定义集。

③ 当 $\alpha_A(Y) = 0$ 时, \tilde{X} 中的全部对象都不能根据 A 中各属性的属性值确定其是否属于 Y , Y 为 A 的完全不可定义集。

当 Y 为 A 的部分可定义集或 Y 为 A 的完全不可定义集时, 称 Y 为 A 的粗糙集。

对例 1 的等价关系 A 有集合 $Y = \{b, c, f\}$ 是粗糙集, 计算集合 Y 的下近似、上近似、正域、负域和边界。

\tilde{X} 中关于 A 的划分为

$$A = \{ \{a, b, c\}, \{d, e\}, \{f\} \}$$

有

$$Y \cap \{a, b, c\} = \{b, c\} \neq \emptyset$$

$$Y \cap \{d, e\} = \emptyset$$

$$Y \cap \{f\} = \{f\} \neq \emptyset$$

可知有

$$A^-(Y) = \{a, b, c\} \cup \{f\} = \{a, b, c, f\}$$

$$A_-(Y) = \{f\}$$

$$POS_A(Y) = A_-(Y) = \{f\}$$

$$BND_A(Y) = A^-(Y) - A_-(Y) = \{a, b, c\}$$

$$NEG_A(Y) = \tilde{X} - A^-(Y) = \{d, e\}$$

8. 相对正域

设决策属性 D 的划分 $Y = (Y_1, Y_2, \dots, Y_M)$, 条件属性 C 相对于决策属性 D 的正域定义为

$$POS(C, D) = \cup C_-(Y_i)$$

9. 决策表的一致性

决策表中的对象 X 按条件属性与决策属性关系看做一条决策规则, 写成

$$\wedge f_{C_i}(X) = f_D(X)$$

式中, C_i 表示多个条件属性; D 表示决策属性; $f_{C_i}(X)$ 表示对象 X 在 C_i 的取值; \wedge 表示逻辑“与”关系。

(1) 一致性决策规则定义

如果对任一个对象 $X_i \neq X_j$, 若条件属性有 $f_{C_i}(X_i) = f_{C_i}(X_j)$, 则决策属性必须有 $f_D(X_i) = f_D(X_j)$, 即一致性决策规则说明条件属性取值相同时, 决策属性取值必须相同。

该定义允许: 若条件属性有 $f_{C_i}(X_i) \neq f_{C_i}(X_j)$, 则决策属性可以是 $f_D(X_i) = f_D(X_j)$ 或 $f_D(X_i) \neq f_D(X_j)$ 。

(2) 决策表一致的定义

在决策表中如果所有对象的决策规则都是一致的, 则该信息表示一致的, 否则信息表示不一致的。在进行属性约简时, 每约简掉一个属性时要检查决策表, 若保持一致性, 则可以删除, 否则不可以删除。

10. 属性约简

(1) 属性依赖度

决策表中决策属性 D 依赖条件属性 C 的依赖度定义为

$$\gamma(C, D) = |POS(C, D)| / |\tilde{X}|$$

式中, $|POS(C, D)|$ 为正域 $POS(C, D)$ 元素的个数, $|\tilde{X}|$ 为整个对象集合的个数。 $\gamma(C, D)$ 的性质如下:

- ① 若 $\gamma = 1$, 表示在已知条件 C 下, 可以将 \tilde{X} 上全部个体分类到决策属性 D 的类别中去。
- ② $\gamma = 0$, 即利用条件 C 不能分类到决策属性 D 的类别中去。
- ③ $0 < \gamma < 1$, 即在已知条件 C 下, 只能将 \tilde{X} 上那些属于正域的个体分类到决策属性 D 的类别中去。

(2) 属性重要度

$C, D \subset A$, C 为条件属性集, D 为决策属性集, $a \in C$, 属性 a 关于 D 的重要度定义为

$$SGF(a, C, D) = \gamma(C, D) - \gamma(C - \{a\}, D)$$

式中, $\gamma(C - \{a\}, D)$ 表示在 C 中缺少属性 a 后, 条件属性与决策属性的依赖程度; $SGF(a, C, D)$ 表示 C 中缺少属性 a 后, 导致不能被准确分类的对象在系统中所占的比例。

$SGF(a, C, D)$ 的性质:

- ① $SGF(a, C, D) \in [0, 1]$ 。
 ② $SGF(a, C, D) = 0$, 表示属性 a 关于 D 是可约简的。
 ③ $SGF(a, C, D) \neq 0$, 表示属性 a 关于 D 是不可约简的。

(3) 最小属性集概念

设 C, D 分别是信息系统 S 的条件属性集和决策属性集, 若属性集 $P (P \subseteq C)$ 是 C 的一个最小属性集, 当且仅当 $\gamma(P, D) = \gamma(C, D)$ 并且 $\forall P' \subset P, \gamma(P', D) \neq \gamma(P, D)$, 则 P 具有与 C 同样的区分决策类的能力。

11. 规则获取

通过分析 \tilde{X} 中的两个划分 $C = \{E_i\}$ 和 $D = \{Y_j\}$ 之间的关系, 把 C 视为分类条件, D 视为分类结论, 可以得到下面的分类规则:

- ① 当 $E_i \cap Y_j \neq \emptyset$ 时, 则有:

$$r_{ij}: Des(E_i) \rightarrow Des(Y_j)$$

$Des(E_i)$ 和 $Des(Y_j)$ 分别是等价集 E_i 和等价集 Y_j 中的特征描述:

- 当 $E_i \cap Y_j = E_i$ 时, 即下近似, 建立的规则 r_{ij} 是确定的, 规则的可信度 $cf = 1$ 。

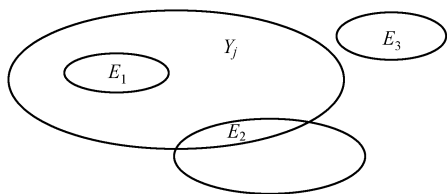


图 8-2 E_i 和 Y_j 的上、下近似关系

- 当 $E_i \cap Y_j \neq E_i$ 时, 即上近似, 建立的规则 r_{ij} 是不确定的, 规则的可信度为

$$cf = \frac{|E_i \cap Y_j|}{|E_i|}$$

如图 8-2 所示, 表示 E_i 和 Y_j 的上、下近似关系。

- ② 当 $E_i \cap Y_j = \emptyset$ 时, E_i 和 Y_j 不能建立规则。

8.2 粗糙集在模式识别中的应用

在科学研究和日常生活中, 人们往往需要根据一些对象的各类属性值, 识别给定对象所属的类型或对观测到的数据进行判别分类, 这是模式识别的主要研究问题。例如, 气象台要通过测量分析最近的气象资料对未来天气类型做出预报, 医生需要根据患者的不同症状检查许多指标以诊断其所患疾病的类型, 手写数字识别系统需要根据手写数字的特征进行分类识别, 等等。

如何在众多资料中选取有代表性的属性, 简化工作, 提高效率; 如何根据给定的属性值给出合理的评价和分类, 都是值得研究的问题。粗糙集理论能够解释不精确数据间的关系, 发现对象和属性间的依赖, 评价属性对分类的重要性, 去除冗余数据, 从而对信息系统进行约简, 因此利用粗糙集理论可以对属性特征进行约简, 略去不必要的属性, 简化数据, 获取决策(分类)规则, 实现模式识别问题中的判别分类功能。

通过下面的实例说明粗糙集理论中的属性约简和规则获取方法, 如表 8-1 所示的是一个决策表样例。

表 8-1 决策表样例

\tilde{X}	C (条件属性)			D (决策属性)
	咳嗽(a)	头晕(b)	发烧(c)	感冒(d)
e_1	是(1)	是(1)	正常(0)	否(0)
e_2	是(1)	是(1)	低烧(1)	是(1)
e_3	是(1)	是(1)	高烧(2)	是(1)
e_4	否(0)	是(1)	正常(0)	否(0)
e_5	否(0)	否(0)	低烧(1)	否(0)
e_6	否(0)	是(1)	高烧(2)	是(1)
e_7	是(1)	否(0)	低烧(1)	是(1)
e_8	否(0)	否(0)	正常(0)	否(0)

1. 等价集、下近似集和依赖度的计算

(1) 条件属性 $C(a, b, c)$ 的等价集

$$E_1\{e_1\}, E_2\{e_2\}, E_3\{e_3\}, E_4\{e_4\}, E_5\{e_5\}, E_6\{e_6\}, E_7\{e_7\}, E_8\{e_8\}$$

(2) 决策属性 $D(d)$ 的等价集

$$Y_1:\{e_1, e_4, e_5, e_8\}, Y_2:\{e_2, e_3, e_6, e_7\}$$

(3) 决策属性的各等价集的下近似集

$$C_Y_1 = \{e_1, e_4, e_5, e_8\}$$

$$C_Y_2 = \{e_2, e_3, e_6, e_7\}$$

(4) 计算 $POS(C, D)$ 和 $\gamma(C, D)$

$$POS(C, D) = C_Y_1 \cup C_Y_2 = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$$

$|POS(C, D)| = 8, |\tilde{X}| = 8, \gamma(C, D) = 1$, 表明在已知条件 $C(a, b, c)$ 下, 可将 \tilde{X} 上全部个体分类到决策属性 D 的类别中。

2. 属性约简

(1) a 的重要度计算

条件属性 $C(b, c)$ 的等价集:

$$E_1\{e_1, e_4\}, E_2\{e_2\}, E_3\{e_3, e_6\}, E_4\{e_5, e_7\}, E_5\{e_8\}$$

决策属性 $D(d)$ 的等价集仍为 Y_1 和 Y_2 。

决策属性的各等价集的下近似集:

$$C_Y_1 = \{e_1, e_4, e_8\}$$

$$C_Y_2 = \{e_2, e_3, e_6\}$$

计算 $POS(C - \{a\}, D)$ 和 $\gamma(C - \{a\}, D)$:

$$POS(C - \{a\}, D) = C_Y_1 \cup C_Y_2 = \{e_1, e_2, e_3, e_4, e_6, e_8\}$$

$$|POS(C - \{a\}, D)| = 6$$

$$\gamma(C - \{a\}, D) = 6/8$$

属性 a 的重要程度:

$SGF(C - \{a\}, D) = \gamma(C, D) - \gamma(C - \{a\}, D) = 1/4 \neq 0$

结论:属性 a 是不可省略的。

(2) b 的重要程度计算

条件属性 C(a,c)的等价集:

$E_1\{e_1\}, E_2\{e_2, e_7\}, E_3\{e_3\}, E_4\{e_4, e_8\}, E_5\{e_5\}, E_6\{e_6\}$

决策属性 D(d)的等价集仍为 Y₁ 和 Y₂。

决策属性的各等价集的下近似集:

$C_{-}Y_1 = \{e_1, e_4, e_5, e_8\}$

$C_{-}Y_2 = \{e_2, e_3, e_6, e_7\}$

计算 POS(C - {b}, D)和 $\gamma(C - \{b\}, D)$:

$POS(C - \{b\}, D) = C_{-}Y_1 \cup C_{-}Y_2 = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$

$|POS(C - \{b\}, D)| = 8$

$\gamma(C - \{b\}, D) = 1$

属性 b 的重要程度:

$SGF(C - \{b\}, D) = \gamma(C, D) - \gamma(C - \{b\}, D) = 0$

结论:属性 b 是可省略的。

(3) c 的重要程度计算

条件属性 C(a,b)的等价集:

$E_1\{e_1, e_2, e_3\}, E_2\{e_4, e_6\}, E_3\{e_5, e_8\}, E_4\{e_7\}$

决策属性 D(d)的等价集仍为 Y₁ 和 Y₂。

决策属性的各等价集的下近似集:

$C_{-}Y_1 = \{e_5, e_8\}$

$C_{-}Y_2 = \{e_7\}$

计算 POS(C - {c}, D)和 $\gamma(C - \{c\}, D)$:

$POS(C - \{c\}, D) = C_{-}Y_1 \cup C_{-}Y_2 = \{e_5, e_7, e_8\}$

$|POS(C - \{c\}, D)| = 3$

$\gamma(C - \{c\}, D) = 3/8$

属性 c 的重要程度:

$SGF(C - \{c\}, D) = \gamma(C, D) - \gamma(C - \{c\}, D) = 5/8 \neq 0$

结论:属性 c 不可省略的。

3. 简化决策表

如表 8-2 所示为简化决策表。

表 8-2 简化决策表

\tilde{X}	咳嗽(a)	发烧(c)	感冒(d)
e_1'	是(1)	正常(0)	否(0)
e_2'	是(1)	低烧(1)	是(1)

续表

\tilde{X}	咳嗽(a)	发烧(c)	感冒(d)
e_3'	是(1)	高烧(2)	是(1)
e_4'	否(0)	正常(0)	否(0)
e_5'	否(0)	低烧(1)	否(0)
e_6'	否(0)	高烧(2)	是(1)

4. 等价集、上下近似集的计算

① 条件属性的等价集:

$$E_1' \{e_1'\}, E_2' \{e_2'\}, E_3' \{e_3'\}, E_4' \{e_4'\}, E_5' \{e_5'\}, E_6' \{e_6'\}$$

② 决策属性 $D(d)$ 的等价集:

$$Y_1' \{e_1', e_4', e_5'\}, Y_2' \{e_2', e_3', e_6'\}$$

5. 获取规则

① $E_1' \cap Y_1' = E_1', E_4' \cap Y_1' = E_4', E_5' \cap Y_1' = E_5'$, 有规则

$$r_{11}: Des(E_1') \rightarrow Des(Y_1'), \text{ 即 } a = 1 \wedge c = 0 \rightarrow d = 0, cf = 1。$$

$$r_{41}: Des(E_4') \rightarrow Des(Y_1'), \text{ 即 } a = 0 \wedge c = 0 \rightarrow d = 0, cf = 1。$$

$$r_{51}: Des(E_5') \rightarrow Des(Y_1'), \text{ 即 } a = 0 \wedge c = 1 \rightarrow d = 0, cf = 1。$$

② $E_2' \cap Y_2' = E_2', E_3' \cap Y_2' = E_3', E_6' \cap Y_2' = E_6'$, 有规则

$$r_{22}: Des(E_2') \rightarrow Des(Y_2'), \text{ 即 } a = 1 \wedge c = 1 \rightarrow d = 1, cf = 1。$$

$$r_{32}: Des(E_3') \rightarrow Des(Y_2'), \text{ 即 } a = 1 \wedge c = 2 \rightarrow d = 1, cf = 1。$$

$$r_{62}: Des(E_6') \rightarrow Des(Y_2'), \text{ 即 } a = 0 \wedge c = 2 \rightarrow d = 1, cf = 1。$$

6. 规则化简

① 对 r_{11} 和 r_{41} 进行合并, 有

$$(a = 0 \vee a = 1) \wedge c = 0 \rightarrow d = 0$$

其中 a 的取值包括了全部取值, 故属性 a 可以删除, 即

$$c = 0 \rightarrow d = 0$$

② 对 r_{32} 和 r_{62} 进行合并, 有

$$(a = 0 \vee a = 1) \wedge c = 2 \rightarrow d = 1$$

同样, 可以删除属性 a , 得到

$$c = 2 \rightarrow d = 1$$

7. 最后的规则

① 发烧 = 正常 \rightarrow 感冒 = 否, 即 $c = 0 \rightarrow d = 0$

② 咳嗽 = 否 \wedge 发烧 = 低烧 \rightarrow 感冒 = 否, 即 $a = 0 \wedge c = 1 \rightarrow d = 0$

③ 发烧 = 高烧 \rightarrow 感冒 = 是, 即 $c = 2 \rightarrow d = 1$

④ 咳嗽 = 是 \wedge 发烧 = 低烧 \rightarrow 感冒 = 是, 即 $a = 1 \wedge c = 1 \rightarrow d = 1$

对于一个条件属性已知, 决策属性未知的新对象, 利用最后获取的规则, 可以得到该对象的决策属性, 实现模式分类。

8.3 粗糙集分类器设计

1. 决策表构造

对于手写数字分类器, 决策表中的条件属性为手写数字特征, 决策属性为手写数字类别。为了更清楚地描述算法, 我们只讨论两类问题, 最终用多个两类判别来实现多类判别。对于两类问题, 决策表中的对象为两类数字的训练样品集合, 每个样品的特征进行二值化作为条件属性值, 二值化方法为: 若特征值为 0, 则对应条件属性值也为 0; 若特征值大于 0, 则对应条件属性值为 1。决策属性值对两类样品分别用 0, 1 表示。利用 200 个训练样品构造决策表, 每类样品各取 100 个, 决策表如表 8-3 所示, 每行代表一个样品, 每列代表一个条件属性, 最后一列为决策属性, $X_1 \sim X_{100}$ 为第一类训练样本, $X_{101} \sim X_{200}$ 为第二类训练样本。

表 8-3 两类问题的决策表

	x_1	x_2	x_3	\cdots	x_{23}	x_{24}	x_{25}	D
X_1	1	0	0	\cdots	0	0	1	0
X_2	1	0	1	\cdots	1	0	1	0
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
X_{100}	1	1	1	\cdots	1	1	1	0
X_{101}	0	0	0	\cdots	0	0	1	1
X_{102}	0	0	1	\cdots	0	0	0	1
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
X_{200}	0	1	0	\cdots	0	1	1	1

2. 规则训练

规则训练是指利用粗糙集理论, 通过对决策表进行条件属性约简、决策规则约简, 获取最小决策规则, 作为最终分类规则。

(1) 等价集、下近似集和依赖度的计算

① 计算条件属性 $X(x_1, x_2, x_3, \cdots, x_{25})$ 的等价集。

② 计算决策属性 D 的等价集。

③ 计算决策属性的各等价集的下近似集 $X _ Y_1, X _ Y_2$ 。

④ 计算 $POS(X, D)$ 和 $\gamma(X, D)$ 。

$$POS(X, D) = X _ Y_1 \cup X _ Y_2, \gamma(X, D) = |POS(X, D)| / |\tilde{X}|$$

(2) 属性约简

对于属性 x_i 计算其重要度:

- ① 计算条件属性 $X - x_i$ 的等价集。
- ② 计算决策属性 D 的等价集。
- ③ 决策属性的各等价集的下近似集。
- ④ 计算 $POS(X - \{x_i\}, D)$ 和 $\gamma(X - \{x_i\}, D)$ 。
- ⑤ 计算属性 x_i 的重要度: $SGF(X - \{x_i\}, D) = \gamma(X, D) - \gamma((X - \{x_i\}), D)$ 。
- ⑥ 如果 $SGF(X - \{x_i\}, D)$ 不等于 0, 则 x_i 不可约简, 否则 x_i 可约简。
- ⑦ 对约简后的决策表进行一致性检查, 如果决策表一致, 则属性可约简; 否则该属性不可约简。
- ⑧ 如果该属性可约简, 则从决策表中删除该属性。

依上述方法对属性 $x_1 \sim x_{25}$ 进行约简, 得到简化后的决策表。

(3) 等价集计算

- ① 计算约简后的条件属性的等价集 $E_1' \sim E_n'$ 。
- ② 计算决策属性 $D(d)$ 的等价集 Y_1', Y_2' 。

(4) 获取规则

- ① 对某一条件属性等价集 E_i' , 如果 $E_i' \cap Y_1' = E_i'$, 则有规则

$$Des(E_i') \rightarrow Des(Y_1')$$

否则如果 $E_i' \cap Y_2' = E_i'$, 则有规则

$$Des(E_i') \rightarrow Des(Y_2')$$

- ② 对每一条件等价集进行规则获取, 保留有效规则。

(5) 规则化简

- ① 对某一条件属性 x_i , 如果有两条规则满足如下条件:
 x_i 分别为 0 和 1, 且除了 x_i 外其他所示条件属性和决策属性都相同。
 则 x_i 属性可以从这两条规则中舍去, 从而实现规则化简。
- ② 对所有属性进行规则化简, 得到最终训练规则。

3. 分类判别

利用训练好的规则, 对待测样品 X , 已知其条件属性 (即特征), 在训练规则中检索, 找到符合规则, 其决策属性即为其类别。

4. 算法流程

利用粗糙集理论进行规则训练和分类判别的流程如图 8-3 所示。

5. 实现步骤

要实现粗糙集多类别分类, 首先要实现两类分类。首先获得任意两类的规则集, 然后对任意待测样品, 分别代入任意两类规则集中进行分类判别, 统计返回类别, 直至代入全部的两类规则集。最终统计类别数最大的类即为待测样品的最终分类。

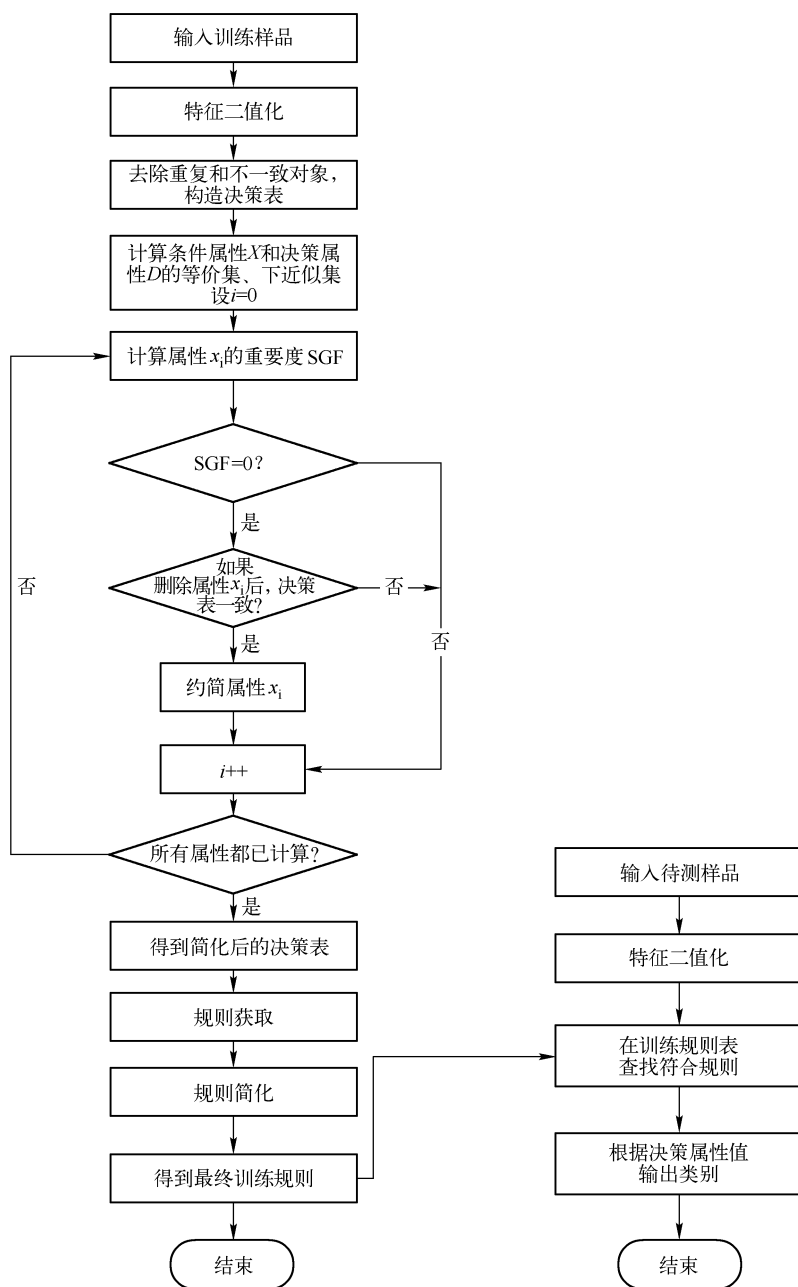


图 8-3 规则训练和分类判别的流程

基于粗糙集理论的两类分类算法的实现步骤如下:

① 提取数字 0~1 两类训练样品的特征值,将其二值化。

② 构造决策表:将 25 个特征作为条件属性 X ,类别作为决策属性 D 。去掉不一致的对象和重复的对象,构造出决策表。图 8-4 为决策表的一部分,列 1~25 为条件属性,即手写数字特征,列 26 为决策属性。每一行对应一训练样品。

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
60	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	0	0	1	1	1	0	0	0	1	1	0
61	1	1	1	1	0	0	1	1	1	0	0	1	1	1	1	0	0	1	1	1	0	0	0	1	1	0
62	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
63	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	1	0	1	1	1	1	0
64	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	0	1	1	1	0	0	1	1	1	0
65	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0
66	0	0	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0
67	0	1	1	1	1	1	1	0	0	1	1	0	0	0	1	1	0	0	0	1	1	1	1	1	1	1
68	1	1	1	1	1	1	0	0	0	1	1	0	0	0	1	1	0	0	1	1	1	1	1	1	1	0
69	1	1	1	1	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	1	1	1	1	1
70	1	1	1	1	1	1	1	0	0	1	1	0	0	0	1	1	0	0	0	1	1	1	1	1	1	1
71	1	1	1	1	1	1	1	0	0	1	1	0	0	0	1	1	0	0	1	1	1	1	1	1	1	0

图 8-4 数字 0 和数字 1 两类决策表的部分数据

③ 计算条件属性 X 的等价集,决策属性 D 的等价集。

④ 计算每个属性的重要程度。重要程度为 0 的直接删除。删除后检查,若决策表不保持一致,则要恢复原来的决策表。最终得到简化后的决策表。如图 8-5 所示的是一个简化后的决策表的一部分,只保留了 2 个条件属性,分别对应两个特征(其列号不表示特征号,实例中 1 列对应第 18 个特征 x_{18} ,2 列对应第 23 个特征 x_{23})。最后一列为决策属性。

	1	2	3
62	1	1	0
63	1	1	0
64	1	1	0
65	1	1	0
66	1	1	0
67	0	1	1
68	0	1	1
69	0	1	1
70	0	1	1

图 8-5 简化后的决策表的部分数据

⑤ 计算条件属性和决策属性的等价集。

⑥ 获取规则。如图 8-6 所示,获取有效规则。

⑦ 简化规则。如图 8-7 所示,简化后的规则为 3 条,Inf 表示规则中的该属性可以简化。如果第 1 列对应第 18 个特征 x_{18} ,第 2 列对应第 23 个特征 x_{23} ,此时规则可描述为

- 如果 x_{23} 为 0,则类别为 0。
- 如果 x_{18} 为 1,则类别为 0。
- 如果 x_{18} 为 0 且 x_{23} 为 1,则类别为 1。

	1	2	3
1	1	0	0
2	1	1	0
3	0	0	0
4	0	1	1

图 8-6 获取有效规则

	1	2	3
1	Inf	0	0
2	1	Inf	0
3	0	1	1

图 8-7 简化后规则

⑧ 输入待测样品,进行规则匹配,实现两类判别。在实际判别中,可能会出现待测样品无


```

bottom(1, patternNum + 1 : patternNum * 2) = 1;
x = [ pattern(class1).feature(:, 1 : patternNum)
      pattern(class2).feature(:, 1 : patternNum); bottom ];
x = ceil(x');
% 一致性检测
ruleNum = patternNum * 2;
% 去重复规则和不一致规则
for m = 1 : ruleNum - 1
    if(m > ruleNum - 1)
        break;
    end
    for n = m + 1 : ruleNum
        if(n > ruleNum)
            break;
        end
        while(x(m, 1 : 25) == x(n, 1 : 25))
            x(n, :) = [];
            ruleNum = ruleNum - 1;
            if(n > ruleNum)
                break;
            end
        end
    end
end

x = x(1 : ruleNum, :);
ruleNumY1 = 0;
for m = 1 : ruleNum
    if(x(m, 26) == 0)
        ruleNumY1 = ruleNumY1 + 1;
    end
end
ruleNumY2 = ruleNum - ruleNumY1;
classX = zeros(ruleNum, ruleNum); % X 属性等价集矩阵
classNum = zeros(ruleNum, 1); % 等价集中元素个数
m = 0; % 等价集个数
n = 1;
% 计算条件 X 等价集
num = 0;
[classX, m, classNum] = CalTiaoJian(num, classX, x, ruleNum, classNum);
% 决策 D 的等价集
classY1 = zeros(1, ruleNumY1);

```

```

classY2 = zeros(1,ruleNumY2);
classY1 = 1;ruleNumY1;
classY2 = ruleNumY1 + 1;ruleNum;
% 决策 D 的下近似集
X _ Y1 = zeros(1,ruleNumY1);
X _ Y2 = zeros(1,ruleNumY2);
[X _ Y1,X _ Y2] = CalXiaJinSi( classX,m,classNum,ruleNumY1);
% 计算 Pos( X,D) 和 r( X,D)
PosXD = [ X _ Y1 X _ Y2];
rXD = size( PosXD,2)/( ruleNum);
% 计算各属性的重要度
import = zeros(1,25);
xReserve = [];
for i = 1:25
    % 计算条件 X - i 的等价集
    classX(:, :) = 0;
    [ classX,m,classNum] = CalTiaoJian( i,classX,x,ruleNum,classNum);
    % 决策 D 的下近似集
    [X _ Y1,X _ Y2] = CalXiaJinSi( classX,m,classNum,ruleNumY1);
    % 计算 Pos( X - i,D) 和 r( X,D)
    PosXiD = [ X _ Y1 X _ Y2];
    import( i) = size( PosXiD,2)/( ruleNum);
    if( rXD - import( i) == 0)
        if( Consistent( i,x,ruleNumY1,ruleNumY2) == 1) % 如果删除后一致
            x(1:ruleNum,i) = 0;
        else
            xReserve = [ xReserve i];
        end
    else
        xReserve = [ xReserve i];
    end
end

% 得到简化后的决策表
xReserve = [ xReserve 26];
xNum = size( xReserve,2);
xNew = zeros( ruleNum,xNum);
for i = 1:ruleNum
    for j = 1:xNum
        xNew( i,j) = x( i,xReserve( j));
    end
end
end
rule = [];

```

```

% 计算条件 X 等价集
num = 0;
classX(:, :) = 0;
[ classX, m, classNum ] = CalTiaoJian( num, classX, xNew, ruleNum, classNum );
% 获取规则
cf = [ ];
for i = 1 : m
    temp = zeros( 1, 2 );
    for j = 1 : classNum( i )
        if( size( find( classY1 == classX( i, j ) ), 2 ) ~ = 0 )
            temp( 1, 1 ) = 1;
        end
        if( size( find( classY2 == classX( i, j ) ), 2 ) ~ = 0 )
            temp( 1, 2 ) = 1;
        end
        if( temp( 1, 1 ) == 1 && temp( 1, 2 ) == 1 )
            cf = [ cf i ]; % 记录 cf 不为 1 的等价集
            break;
        end
    end
end

end

temp = size( cf, 2 );
if( temp ~ = 0 ) % 舍去 cf 不为 1 的规则
    for i = 1 : temp
        for j = 1 : classNum( cf( i ) )
            xNew( classX( cf( i ), j ), xNum ) = 2;
        end
    end
end

for i = 1 : ruleNum
    while( x( i, xNum ) == 2 )
        xNew( i, : ) = [ ];
        ruleNum = ruleNum - 1;
        if( i > ruleNum )
            break;
        end
    end
    if( i > = ruleNum )
        break;
    end
end

end

% 简化规则表, 去掉重复规则

```

```

for m = 1 : ruleNum - 1
    if( m > ruleNum - 1 )
        break;
    end
    for n = m + 1 : ruleNum
        if( n > ruleNum )
            break;
        end
        while( xNew( m, 1 : xNum ) == xNew( n, 1 : xNum ) )
            xNew( n, : ) = [ ] ;
            ruleNum = ruleNum - 1 ;
            if( n > ruleNum )
                break;
            end
        end
    end
end

ruleNumY1 = 0;
for m = 1 : ruleNum
    if( xNew( m, xNum ) == 0 )
        ruleNumY1 = ruleNumY1 + 1 ;
    end
end

ruleNumY2 = ruleNum - ruleNumY1 ;
for m = 1 : ruleNum
    a = [ ] ;
    for j = 1 : xNum
        a = [ a xNew( m, j ) ] ;
    end
    rule = [ rule; a ] ;
end

% 规则化简
oldRule = rule;
newRule = [ ] ;
ruleJ = [ ] ; % 统计可化简的规则
for i = 1 : size( xReserve, 2 )
    rule = oldRule;
    rule( :, i ) = 0;
    flag = false;
    for m = 1 : ruleNumY1 - 1
        for n = m + 1 : ruleNumY1

```

```

        if( rule( m, :) == rule( n, :) ) % 可化简
            rule( m, i) = inf;
            newRule = [ newRule; rule( m, :) ] ;
            flag = true;
            ruleJ = [ ruleJ m n ] ;
            break;
        end
    end
end
if( flag)
    break;
end
end
flag = false;
for m = ruleNumY1 + 1 ; ruleNum - 1
    for n = m + 1 ; ruleNum
        if( rule( m, :) == rule( n, :) ) % 可化简
            rule( m, i) = inf;
            newRule = [ newRule; rule( m, :) ] ;
            flag = true;
            ruleJ = [ ruleJ m n ] ;
            break;
        end
    end
end
if( flag)
    break;
end
end
end

for i = 1 ; ruleNum
    b = size( find( ruleJ == i ), 2 ) ;
    if( b == 0 ) % 该规则不可约简
        newRule = [ newRule; oldRule( i, :) ] ;
    end
end
newRule = [ newRule; xReserve ] ;

```

% 函数名称: CalTiaoJian()

% 参数: num: 条件属性号; classX: 条件等价集; x: 决策表;

% ruleNum: 规则数; classNum: 条件等价集中元素数

% 返回值: classX: 条件等价集; m: 等价集个数; classNum: 条件等价集中元素数

% 函数功能: 计算条件等价集


```

for j = 1:classNum(i)
    if( classX(i,j) > ruleNumY1) % 不属于 X_ Y1;
        Y1 = false;
    else
        Y2 = false;
    end
    if( ~ Y1&&~ Y2)
        break;
    end
end
if( Y1)
    X_ Y1 = [ X_ Y1 classX(i,1:classNum(i)) ];
elseif( Y2)
    X_ Y2 = [ X_ Y2 classX(i,1:classNum(i)) ];
end
end

%% 函数名称:Consistent( )
% 参数:    num:条件属性号;x:决策表;ruleNumY1:决策属性 1 的等价集中的元素数;
%          ruleNumY2:决策属性 2 的等价集中的元素数
% 返回值:    cons:一致性变量;如果 cons 为 0,则不一致,否则一致
% 函数功能:一致性检测

function [ cons ] = Consistent( num,x,ruleNumY1,ruleNumY2)

    cons = 1;
    x( :,num) = 0;
    for i = 1:ruleNumY1
        for j = ruleNumY1 + 1:ruleNumY1 + ruleNumY2
            if( x(i,1:25) == x(j,1:25) )
                cons = 0;
            end
        end
    end
end
end

```

(2) 粗糙集分类程序代码

函数调用关系

```

CuCao() %粗糙集分类
└─> result=CuCao2Class(class1,class2,sample,rule) %粗糙集两类分类

```

```

% 函数名称:CuCao( )
% 参数:sample:待测样品
% 返回值:result:分类结果
% 函数功能:粗糙集分类
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function result = CuCao( sample );

    load ruleSrtuct;% 读取已训练规则
    num = zeros( 1,10 );
    classnum = 0;
    for i = 1:10
        for j = 1:i - 1
            % 两类判别
            G = CuCao2Class( i,j,sample,ruleSrtuct( i,j). rule );
            if( G == 0)
                num( i ) = num( i ) + 1;
            elseif( G == 1 )
                num( j ) = num( j ) + 1;
            end
        end
    end
    [ max _ val,max _ pos ] = max( num );
    result = max _ pos - 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 函数名称:CuCao2Class( )
% 参数:class1:类号 1;class2:类别 2;sample:待测样品;rule:训练规则
% 返回值: result:分类结果
% 函数功能:粗糙集两类分类
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function result = CuCao2Class( class1 ,class2,sample,rule );

    sample = ceil( sample );% 测试样品二值化
    ruleNum = size( rule,1 ) - 1;% 规则数
    xNum = size( rule,2 ) - 1;% 属性数
    result = - 1;
    for i = 1:ruleNum % 匹配规则
        flag = true;
        for j = 1:xNum
            if( rule( i,j) ~ = 100&&sample( rule( ruleNum + 1 ,j) ) ~ = rule( i,j) )
                flag = false;
                break;
            end
        end
        if( flag)

```

```

        result = rule(i,xNum + 1);
        break;
    end
end
% 找最相近的规则
if(result == -1)
    ruleSel = zeros(1,ruleNum);
    for i = 1:ruleNum
        for j = 1:xNum
            if( rule(i,j) ~ = 100&&sample( rule( ruleNum + 1,j)) ~ = rule(i,j))
                ruleSel(i) = ruleSel(i) + 1;
            end
        end
    end
    [ a b ] = min( ruleSel );
    result = rule( b,xNum + 1 );
end

```

7. 效果图

首先单击粗糙集训练菜单,进行规则训练,训练完成后输出对话框提示,如图 8-8(a)所示。然后手写一个数,单击粗糙集分类菜单,提示分类结果,如图 8-8(b)和(c)所示。



(a) 规则训练



(b) 手写数字



(c) 粗糙集分类

图 8-8 粗糙集分类算法效果

本章小结

本章介绍了粗糙集理论的基本概念,包括知识系统、决策表、集合的上下近似集、粗糙集的

定义以及属性约简和规则化简理论等。还通过实例介绍了粗糙集理论在模式识别中的应用。最后详细介绍了基于粗糙集理论的分类方法,包括分类器设计、算法流程、实现步骤等。

习题 8

- 1. 简述决策表、等价集以及粗糙集的概念。
- 2. 简述属性约简和规则化简方法。
- 3. 如表 8-4 所示,已知购买计算机意向决策表,利用粗糙集理论对该表进行属性约简和规则化简,并获取最终规则。

表 8-4 购买计算机意向决策表

	C(条件属性)				D(决策属性)
\tilde{X}	年龄(c_1)	学历(c_2)	收入(c_3)	信用度(c_4)	购买计算机(d)
e_1	<30(0)	本科以上(1)	低(0)	高(1)	不买(0)
e_2	<30(0)	本科以下(0)	低(0)	低(0)	不买(0)
e_3	<30(0)	本科以上(1)	高(1)	低(0)	买(1)
e_4	<30(0)	本科以下(0)	高(1)	高(1)	买(1)
e_5	30~60(1)	本科以上(0)	低(0)	高(1)	买(1)
e_6	30~60(1)	本科以上(1)	高(1)	高(1)	买(1)
e_7	30~60(1)	本科以下(0)	高(1)	低(0)	买(1)
e_8	>60(2)	本科以上(1)	低(0)	高(1)	买(1)
e_9	>60(2)	本科以下(0)	低(0)	低(0)	不买(0)
e_{10}	>60(2)	本科以上(1)	高(1)	低(0)	不买(0)

- 4. 叙述粗糙集理论在分类问题中的实现方法和步骤。

第 9 章 聚 类 分 析

本章要点:

- ☑ 聚类的设计
- ☑ 基于试探的未知类别聚类算法
- ☑ 层次聚类算法
- ☑ 动态聚类算法
- ☑ 模拟退火聚类算法

9.1 聚类的设计

聚类分析是指事先不了解一批样品中的每一个样品的类别或者其他的先验知识,而唯一的分类根据是样品的特征,利用某种相似性度量的方法,把特征相同或相近的归为一类,实现聚类划分。例如,对于一幅手写数字图像,如图 9-1 所示,将相同的手写数字划分为一类,即聚类分析要解决的问题。

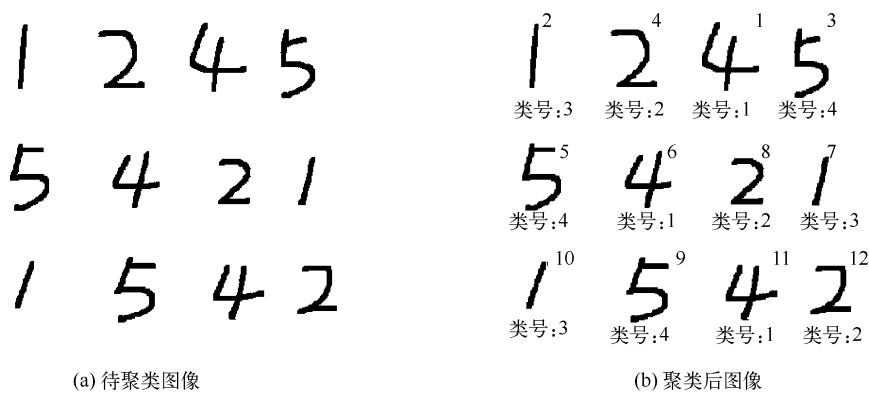


图 9-1 手写数字图像聚类

本书从第 9 章到第 13 章对各种聚类算法进行了理论分析和实例介绍,为了便于读者对后面章节的阅读,本节将对聚类算法设计中的结构定义以及距离计算方法等常用函数功能进行介绍,使读者能够更清楚地阅读算法程序。

1. 样品结构设计

样品是聚类分析中的最基本单位,例如,手写数字聚类中每一个手写数字即为一个样品,通常一个样品的结构包括样品特征和所属类别两部分。

在本书基于 Matlab 的聚类算法设计中,样品结构定义如下。

(1) 样品集 `m_pattern`

样品集为多个样品的集合,一个具有 N 个样品的样品集 `m_pattern`,定义为 `m_pattern =`

$\{m_pattern(1), m_pattern(2), \dots, m_pattern(N)\}$, 其中 $m_pattern$ 中每一个元素为一个样品结构。

(2) 样品 $m_pattern(i)$

对于样品集中的样品 $m_pattern(i)$, 其结构定义为:

```
Struct m_pattern(i)
{
    feature;
    category;
}
```

其中 **feature** 是该样品的特征矩阵, 本书中对每个样品划分成 7×7 块, 共 49 个特征。category 为样品所属类别。

2. 聚类中心结构设计

聚类中心是指当对样品进行聚类划分之后, 对每一个划分好的类用一个结构来描述, 这个结构就是聚类中心。聚类中心结构包括聚类中心特征, 属于该类的样品数目, 类索引值。

类似于样品结构定义, 聚类中心结构定义如下。

(1) 聚类中心集 m_center

聚类中心集为多个聚类中心的集合, 一个具有 M 个类的聚类中心集 m_center , 定义为 $m_center = \{m_center(1), m_center(2), \dots, m_center(M)\}$, 其中 m_center 中每一个元素为一个聚类中心结构。

(2) 聚类中心 $m_center(i)$

聚类中心 $m_center(i)$, 其结构定义为

```
Struct m_center(i)
{
    feature;
    patternNum;
    index;
}
```

其中, **feature** 是该聚类中心的特征矩阵, patternNum 为属于该类的样品数目, index 为类的索引号。

3. 样品(或聚类中心)与样品(或聚类中心)的距离

本书中计算样品(或聚类中心)特征之间距离有四种方法, 分别是欧氏距离法、夹角余弦距离法、二值夹角余弦法和具有二值特征的 Tanimoto 测度。计算公式见本书第 3 章的表 3-1。

本书中距离计算的 Matlab 函数为: GetDistance(), 具体的函数说明及代码如下:

```
% %%%%%%%%%%
% 函数名称      GetDistance( pattern1, pattern2, type )
% 参数          pattern1: 样品(或聚类中心)1 结构
```

```

%          pattern2: 样品(或聚类中心)2 结构
%          type:距离模式 1:欧氏距离;2:夹角余弦距离; 3:特征是二值时
%              的夹角余弦距离;4:具有二值特征的 Tanimoto 测度
% 返回值          result:距离
% 函数功能          计算样品(或聚类中心)1 和样品(或聚类中心)2 间的距离,距离模式
%                  由参数 type 给定
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ result ] = GetDistance( pattern1 ,pattern2 ,type )
    result = 0;
    global Nwidth;% 特征矩阵的宽度,本书取 7,即样品特征为 7 * 7 矩阵
    switch( type)
    case 1 % 欧式距离
        result = sum( ( pattern1. feature( : ) - pattern2. feature( : ) ). ^2 );
        result = sqrt( result );
    case 2 % 夹角余弦
        a = 0;
        b1 = 0;
        b2 = 0;
        for i = 1 : Nwidth
            for j = 1 : Nwidth
                a = a + pattern1. feature( i ,j ) * pattern2. feature( i ,j );
                b1 = b1 + pattern1. feature( i ,j ) * pattern1. feature( i ,j );
                b2 = b2 + pattern2. feature( i ,j ) * pattern2. feature( i ,j );
            end
        end
        if( b1 * b2 ~ = 0)
            result = 1 - a/sqrt( b1 * b2 );
        else
            result = - 1 ;
        end
    case 3 % 二值夹角余弦
        t1 = zeros( Nwidth, Nwidth );
        t2 = zeros( Nwidth, Nwidth );
        a = 0;
        b1 = 0;
        b2 = 0;
        for i = 1 : Nwidth
            for j = 1 : Nwidth
                if( pattern1. feature( i ,j ) > 0. 2 )
                    t1( i ,j ) = 1 ;
                end
                if( pattern2. feature( i ,j ) > 0. 2 )
                    t2( i ,j ) = 1 ;
                end
                a = a + t1( i ,j ) * t2( i ,j );
                b1 = b1 + t1( i ,j ) * t1( i ,j );
                b2 = b2 + t2( i ,j ) * t2( i ,j );
            end
        end
    end
end

```



```

function [ m_center_i ] = CalCenter( m_center_i, m_pattern, patternNum )
    global Nwidth;
    temp = zeros( Nwidth, Nwidth ); % 临时存储中心的特征值
    a = 0; % 记录该类中元素个数
    for i = 1 : patternNum
        if ( m_pattern(i).category == m_center_i.index ) % 累加中心所有样品
            a = a + 1;
            temp = temp + m_pattern(i).feature;
        end
    end
    m_center_i.patternNum = a;
    if (a ~= 0)
        m_center_i.feature = temp/a; % 取均值
    else
        m_center_i.feature = temp;
    end
end

```

9.2 基于试探的未知类别聚类算法

定义误差平方和为

$$J = \sum_{i=1}^M \sum_{X \in \omega_i} \| X - \overline{X^{(\omega_i)}} \|^2 \quad (9-1)$$

式中, M 是聚类中心的个数, M 应该小于样品的总个数。 ω_i 表示第 i 类, $\overline{X^{(\omega_i)}}$ 表示第 i 类的聚类中心向量。针对所有样品假设某种聚类方案, 计算 J 值, 找到 J 值最小的那一种聚类方案, 则认为该方法为最优聚类。以下讨论的是在聚类数未知情况下, 以该准则为聚类方案。

9.2.1 最临近规则的试探法

1. 理论基础

设有 N 个样品: X_1, X_2, \dots, X_N , 并选取任一非负的阈值 T 。为方便起见, 我们假设前 i ($i < N$) 个样品已经被分到 k ($k \leq i$) 个类中。则第 $i+1$ 个样品应该归入哪一个类中呢? 假设归入 ω_a 类, 要使 J 最小, 则应满足 $|X_{i+1} - \overline{X^{(\omega_a)}}| \leq |X_{i+1} - \overline{X^{(\omega_b)}}|$ ($1 \leq b \leq k$)。若 X_{i+1} 到 ω_a 类的距离大于给定的阈值 T , 即 $|X_{i+1} - \overline{X^{(\omega_a)}}| > T$, 则应为 X_{i+1} 建立一个新的类 ω_{k+1} 。在未将所有的样品分类前, 类数是不能确定的。

这种算法与第一个中心的选取、阈值 T 的大小、样品排列次序以及样品分布的几何特性有关。这种方法运算简单, 若有关于模式几何分布的先验知识做指导给出阈值 T 及初始点, 则能较快地获得合理的聚类结果。

合理地选择聚类中心和阈值, 将会得到正确的聚类结果, 如图 9-2(a) 所示, 若选择的中心和阈值不当, 得到聚类结果比较粗糙, 甚至错误, 如图 9-2(b) 和(c) 所示。


```

% 函数名称:C_ZuiLinJin()
% 参数:m_pattern:样品特征库;patternNum:样品数目
% 返回值:m_pattern:样品特征库
% 函数功能:按照最临近规则对全体样品进行分类
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ m_pattern ] = C_ZuiLinJin( m_pattern,patternNum )
    global Nwidth;
    m_center(1).feature = m_pattern(1).feature;% 将第一个样品作为第一个中心
    m_center(1).index = 1;
    m_center(1).patternNum = 1;
    m_pattern(1).category = 1;
    centerNum = 1;

    m_min = inf;
    m_max = 0;
    disType = DisSelDlg( );
    sum = 0;
    div = 0;
    T = InputThreshDlg( m_pattern,patternNum,disType );% 获得阈值
    for i = 1:patternNum
        centerdistance = inf;
        index = 1;
        for j = 1:centerNum
            dis = GetDistance( m_pattern(i),m_center(j),disType );
            if( dis < centerdistance )
                centerdistance = dis;
                index = j;
            end
        end
        if ( centerdistance < T )% 距离小于阈值则将样品归入该类
            m_pattern(i).category = m_center(index).index;
            m_center(index) = CalCenter( m_center(index),m_pattern,patternNum );
        else % 新建聚类中心
            centerNum = centerNum + 1;
            m_pattern(i).category = centerNum;
            m_center(centerNum).feature = m_pattern(i).feature;
            m_center(centerNum).index = centerNum;
        end
    end
end

```

4. 效果图

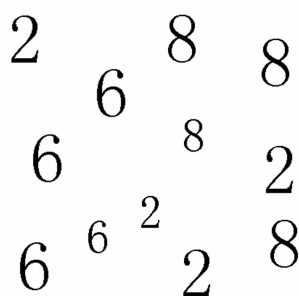
如图 9-3(a) 所示的是待聚类的原始图,输入阈值为 1.5,如图 9-3(b) 所示,经过程序运算

得到了聚类结果,如图 9-3(c)所示,每个物体的右上角标号是该物体的标号,左下角是该物体的所属类号,从效果图上可以看出,样品被正确分类;当阈值为 2.5,过大的时候,如图 9-3(d)所示,不同样品被归入同一类,如图 9-3(e)所示。除了对数字进行聚类分析,还可以对几何图形进行聚类,如图 9-3(f)、(g)、(h)所示。

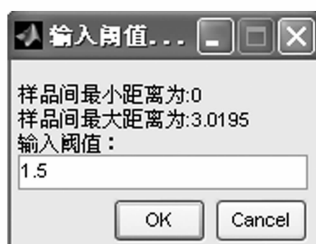
最临近规则的试探法受到阈值 T 的影响很大。阈值的选取是分类成败的关键之一。用户可以根据对话框中给出的参考值确定阈值,一般应介于最大值和最小值之间。

这种算法与第一个中心的选取、阈值 T 的大小、样品排列次序以及样品分布的几何特性有关。

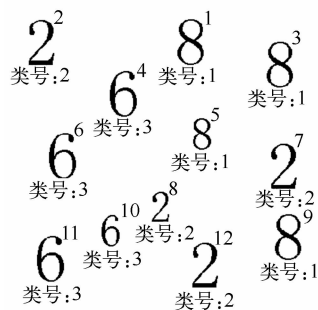
不难看出,这种方法运算简单,若有关于模式几何分布的先验知识做指导给出阈值 T 及初始点,则能较快地获得合理的聚类结果。



(a) 原始数据



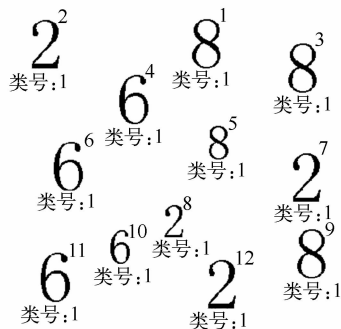
(b) 输入阈值



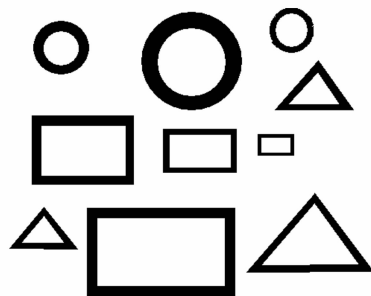
(c) 正确输出结果



(d) 输入阈值过大



(e) 不同样品被归入同一类

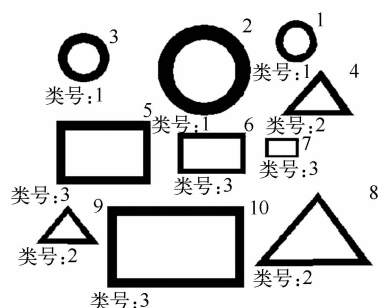


(f) 几何图形原始图像

图 9-3 最临近规则聚类效果图



(g) 输入阈值



(h) 对几何图形的最临近规则聚类结果

图 9-3 最临近规则聚类效果图(续)

9.2.2 最大最小距离算法

1. 理论基础

最大最小距离算法充分利用样品内部特性,计算出所有样品间的最大距离 \maxdistance 作为归类阈值的参考,改善了分类的准确性。若某样品到某一个聚类中心的距离小于 $\maxdistance/3$,则归入该类,否则建立新的聚类中心。

2. 实现步骤

① 选一个样品作为第一个聚类中心 $m_center(1)$,以第一个样品的特征值作为第一个中心的特征值,当前类中心的数目为 1。

$m_center(1).feature = m_pattern(1).feature, m_center(1).index = 1$,不妨令

$m_pattern(1).category = 1$ 。 $centerNum$ 记录当前类的数目, $centerNum = 1$ 。

② 查找离 $m_center(1)$ 最远的样品 $m_pattern(i)$,设最大距离为 \maxdistance 。令最远的样品 $m_pattern(i)$ 为第二个类,增加一个中心个数 $centerNum++$; $m_center(2).feature = m_pattern(i).feature$; $m_pattern(i).category = 2$ 。

③ 逐个计算其余各样品 $m_pattern(i)$ 到各个聚类中心 $m_center(j)$ ($1 \leq j \leq centerNum$) 间的距离,查看样品 $m_pattern(i)$ 距离哪一个中心近,找出最近的中心为 $m_center(index)$,计算其距离为 $tDistance$ 。

若 $tDistance \leq \maxdistance/3$,则将该样品 $m_pattern(i)$ 归入距离最近的类,即 $m_pattern(i).category = m_center(index).index$;

重新计算序号为 $index$ 的中心特征值。

若 $tDistance > \maxdistance/3$,则以该样品为中心建立新的聚类中心;增加一个中心个数, $centerNum++$,即 $m_center(centerNum) = m_pattern(i)$, $m_pattern(i).category = centerNum$ 。

④ 重复步骤③,直到所有样品分类完毕。

3. 编程代码

```
%%%%%%%%%%
% 函数名称:C_ZuiDaZuiXiaoJuLi()
% 参数:    pattern:样品特征库; patternNum:样品数目
```

```

% 返回值: pattern:样品特征库
% 函数功能: 按照最大最小距离规则对全体样品进行分类
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ m_pattern ] = C_ZuiDaZuiXiaoJuLi( m_pattern, patternNum )
    disType = DisSelDlg(); % 获得距离计算类型
    maxDistance = 0; % 记录两类间的最大距离,用做类分割阈值
    index = 1; % 记录距离第一个中心最远的样品

    m_center(1).feature = m_pattern(1).feature; % 第一个聚类中心
    m_center(1).index = 1;
    m_center(1).patternNum = 1;
    m_pattern(1).category = 1;

    for i = 1:patternNum % 第二个聚类中心
        tDistance = GetDistance( m_pattern(i), m_center(1), disType );
        if( maxDistance < tDistance )
            maxDistance = tDistance;
            index = i;
        end
    end
    m_center(2).feature = m_pattern(index).feature;
    m_center(2).index = 2;
    m_pattern(index).category = 2;
    centerNum = 2;

    for i = 1:patternNum
        MAX = inf;
        index = 0; % 记录样品距离最近的中心
        for j = 1:centerNum
            tDistance = GetDistance( m_pattern(i), m_center(j), disType );
            if( MAX > tDistance )
                MAX = tDistance;
                index = j;
            end
        end
        if( MAX > maxDistance/3 ) % 样品到最近中心的距离大于阈值,建立新的聚类中心
            centerNum = centerNum + 1;
            m_center( centerNum ).feature = m_pattern(i).feature;
            m_center( centerNum ).index = centerNum;
            m_pattern(i).category = centerNum;
        else % 归入 index 类中
            m_pattern(i).category = m_center(index).index;
            CalCenter( m_center(index), m_pattern, patternNum );
        end
    end
end

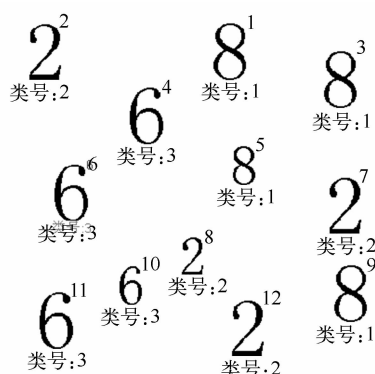
```

4. 效果图

每个物体的右上角标号是该物体的标号,左下角是该物体的所属类号。最大最小距离法不需要用户输入聚类阈值,只需要选择距离计算类型即可,如图9-4(a)、(c)所示为选择距离计算公式,程序会根据所有样品数据自动确定阈值。若样品集中没有突出的噪声,程序一般会正确聚类,如图9-4(b)所示。然而,样品中含有离中心较远的孤立点时,聚类阈值会受到干扰。该算法对图形数据聚类的效果如图9-4(d)所示。



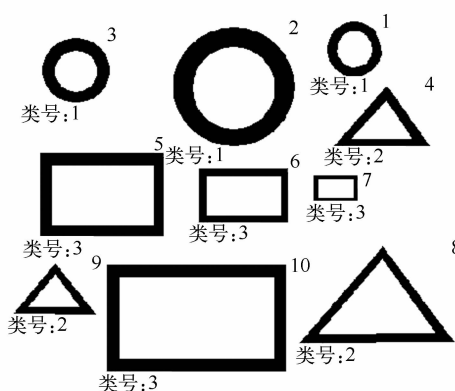
(a) 选择欧氏距离



(b) 数字聚类结果



(c) 采用夹角余弦距离模式



(d) 图形聚类结果

图9-4 最大最小距离法聚类效果图

9.3 层次聚类算法

与未知类别的聚类算法不同,层次聚类分为合并算法和分裂算法。合并算法会在每一步减小聚类中心数量,聚类产生的结果来自于前一步的两个聚类的合并;分裂算法与合并算法的原理相反,在每一步增加聚类中心数量,每一步聚类产生的结果,都是将前一步的一个聚类中心分裂成两个得到的。合并算法,先将每个样品自成一类,然后根据类间距离的不同,合并距离小于阈值的类。

① 设有 N 个样品,这里取 $N=4$ 。每个样品自成一类,计算各类间的距离填入表9-1。

初始距离计算公式为 $D_{ij} = \|\omega_i - \omega_j\| = \|X_i - X_j\|$ 。

表 9-1 聚类中心间的距离表

	ω_1	ω_2	ω_3	ω_4
ω_1	D_{11}	D_{12}	D_{13}	D_{14}
ω_2	D_{21}	D_{22}	D_{23}	D_{24}
ω_3	D_{31}	D_{32}	D_{33}	D_{34}
ω_4	D_{41}	D_{42}	D_{43}	D_{44}

② 求表 9-1 中最小的值, 设为 D_{ij} , 即距离最近的两类是 ω_i, ω_j , 若 $D_{ij} < T$, 合并 ω_i, ω_j 类。如 $D_{34} < T$, 则合并 ω_3 与 ω_4 , 组成新的类 $\omega_{3,4}$ 。

③ 下一步要确定各类到 $\omega_{3,4}$ 的距离, 填写表 9-2。这里介绍了最短距离法、最长距离法、中间距离法、重心法、类平均距离法。各种算法的计算公式如表 9-3 所示。

表 9-2 合并后的聚类中心间距

	ω_1	ω_2	$\omega_{3,4}$
ω_1	D_{11}	D_{12}	D_{13}
ω_2	D_{21}	D_{22}	D_{23}
ω_3	$D_{34,1}$	$D_{34,2}$	$D_{34,3}$

表 9-3 两类合并后到其他类间的距离计算公式

距离计算方法	ω_j 是由 ω_m, ω_n 两类合并而成的, 定义类 ω_i 与 ω_j 类的距离为: $D_{i,j}$	ω_i 中有 N_i 个样品, ω_m 中有 N_m 个样品, ω_n 中有 N_n 个样品
最短距离法	$D_{i,j}$ 为 ω_i 类中所有样品与 ω_j 类中所有样品间的最小距离	$D_{i,j} = D_{i,mn} = \min(D_{i,m}, D_{i,n})$
最长距离法	$D_{i,j}$ 为 ω_i 类中所有样品与 ω_j 类中所有样品间的最长距离	$D_{i,j} = D_{i,mn} = \max(D_{i,m}, D_{i,n})$
中间距离法	它介于最长距离与最短距离之间, 其中 $D_{i,m}, D_{i,n}, D_{m,n}$ 可以用类间的最长距离或最短距离计算	$D_{i,j} = D_{i,mn} = \left(\frac{1}{2} D_{i,m}^2 + \frac{1}{2} D_{i,n}^2 - \frac{1}{4} D_{m,n}^2 \right)^{\frac{1}{2}}$
重心法	重心法考虑了类内样品数目对类间距离的影响	$D_{i,j} = D_{i,mn} = \left(\frac{N_m}{N_m + N_n} D_{i,m}^2 + \frac{N_n}{N_m + N_n} D_{i,n}^2 - \frac{N_m N_n}{N_m + N_n} D_{m,n}^2 \right)^{\frac{1}{2}}$
平均距离法	用两类内所有样品距离的平均值作为两类的距离	$D_{i,j} = D_{i,mn} = \left(\frac{N_m}{N_m + N_n} D_{i,m}^2 + \frac{N_n}{N_m + N_n} D_{i,n}^2 \right)^{\frac{1}{2}}$

注: 其中 $D_{i,m}, D_{i,n}$ 定义见本书第 3 章的表 3-2。

9.3.1 最短距离法

1. 理论基础

最短距离法认为, 只要两类的最小距离小于阈值, 就将两类合并成一类。定义 $D_{i,j}$ 为 ω_i 类

中所有样品和 ω_j 类中所有样品间的最小距离,即

$$D_{i,j} = \min \{ d_{UV} \}$$

式中, d_{UV} 表示 ω_i 类中的样品 U 与 ω_j 类中的样品 V 之间的距离。若 ω_j 类是由 ω_m, ω_n 两类合并而成的,则

$$\begin{aligned} D_{i,m} &= \min \{ d_{UA} \} & U \in \omega_i \text{ 类}, A \in \omega_m \text{ 类} \\ D_{i,n} &= \min \{ d_{UB} \} & U \in \omega_i \text{ 类}, B \in \omega_n \text{ 类} \end{aligned}$$

递推可得

$$D_{i,j} = \min \{ D_{i,m}, D_{i,n} \} \quad (9-2)$$

例如:计算 ω_1 到 $\omega_{3,4}$ 的距离 $D_{1,34}$,先计算 ω_1 类中各个样品到 ω_3 类中各个样品的距离,取最小值为 $D_{1,3}$;然后计算 ω_1 类中各个样品到 ω_4 类中各个样品的距离,取最小值为 $D_{1,4}$;取 $D_{1,3}$ 、 $D_{1,4}$ 中的最小值为 $D_{1,34}$ 。

2. 实现步骤

- ① 获得所有样品特征。
- ② 输入阈值 T (计算所有样品距离的最大值与最小值,输出,作为阈值的参考)。
- ③ 将所有样品各分一类,聚类中心数 $\text{centerNum} = \text{样品总数}$ $\text{patternNum}, m_pattern(i).$
 $\text{category} = i; m_center(i). \text{feature} = m_pattern(i). \text{feature}。$

④ 对所有样品循环:

- 找到距离最近的两类 p_i, p_j , 设距离为 minDis 。
- 若 $\text{minDis} \leq T$, 则合并 p_i, p_j , 将类号大的归入到类号小的类中, 调整其他类保持类号连续。否则 $\text{minDis} > T$, 即两类间的最小距离大于阈值, 则退出循环。

3. 编程代码

```
% 函数名称: C_ZuiDuanJuLi()
% 参数: m_pattern: 样品特征库; patternNum: 样品数目
% 返回值: m_pattern: 样品特征库
% 函数功能: 按照最短距离法对全体样品进行分类
function [ m_pattern ] = C_ZuiChangJuLi( m_pattern, patternNum )
    disType = DisSelDlg(); % 获得距离计算类型
    T = InputThreshDlg( m_pattern, patternNum, disType ); % 获得阈值
    % 初始化, 所有样品各分一类
    for i = 1 : patternNum
        m_pattern(i).category = i;
    end

    while( true )
        minDis = inf;
        pi = 0;
```

```

pj=0;
%寻找距离最近的两类 pi,pj,记录最小距离 minDis
for i=1:patternNum-1
    for j=i+1:patternNum
        if(m_pattern(i).category~=m_pattern(j).category)
            tempDis=GetDistance(m_pattern(i),m_pattern(j),disType);
            if(tempDis<minDis)
                minDis=tempDis;
                pi=m_pattern(i).category;
                pj=m_pattern(j).category;
            end
        end
    end
end

if(minDis<=T)%距离小于阈值,合并 pi,pj 类
    if(pi>pj)%将较大类号归入较小类号
        temp=pi;
        pi=pj;
        pj=temp;
    end
    for i=1:patternNum
        if(m_pattern(i).category==pj)
            m_pattern(i).category=pi;
        elseif(m_pattern(i).category>pj)
            m_pattern(i).category=m_pattern(i).category-1;
        end
    end
else
    break;
end
end
end

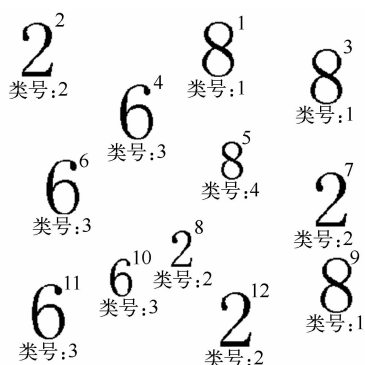
```

4. 效果图

在最短距离法中,只要两类的最近距离在阈值之内,就会被合并成一个类,聚类效果如图 9-5(a) 和(b)所示,阈值过小,同类样品将被归入不同的类中;如图 9-5(c)和(d)所示,阈值恰当,被正确聚类;如图 9-5(e)和(f)所示为二值夹角余弦距离模式下,图形数据聚类结果。对相同的数据,要想获得同样的聚类效果,最短距离法的阈值要比其他算法大一些。



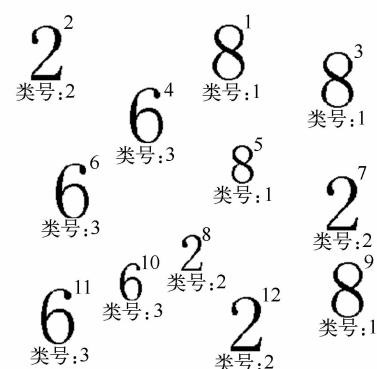
(a) 阈值过小



(b) 同类样品被归入不同的类中



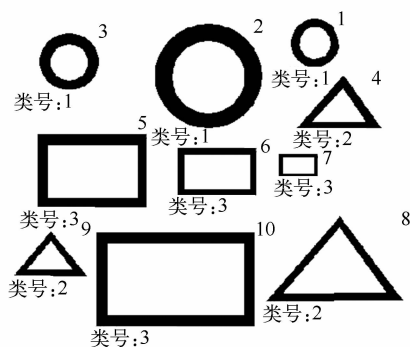
(c) 阈值恰当



(d) 正确聚类



(e) 二值夹角余弦距离模式



(f) 图形数据聚类结果

图 9-5 最短距离法聚类效果图

9.3.2 最长距离法

1. 理论基础

在该算法中,两类中的所有样品间的距离必须都小于阈值,才能将两类合并。定义 $D_{i,j}$ 为

ω_i 类中所有样品与 ω_j 类中所有样品间的最大距离,则有

$$D_{i,j} = \max \{ d_{UV} \}$$

其中 d_{UV} 表示 ω_i 类中的样品 U 与 ω_j 类中的样品 V 之间的距离。

若 ω_j 类由 ω_m, ω_n , 两类合并而成,则

$$D_{i,m} = \max \{ d_{UA} \} \quad U \in \omega_i \text{ 类}, A \in \omega_m \text{ 类}$$

$$D_{i,n} = \max \{ d_{UB} \} \quad U \in \omega_i \text{ 类}, B \in \omega_n \text{ 类}$$

递推可得

$$D_{i,j} = \max \{ D_{i,m}, D_{i,n} \} \quad (9-3)$$

例如:计算 ω_1 到 $\omega_{3,4}$ 的距离 $D_{1,34}$,先计算 ω_1 类中各个样品到 ω_3 类中各个样品的距离最大值为 $D_{1,3}$;然后计算 ω_1 类中各个样品到 ω_4 类中各个样品的距离最大值为 $D_{1,4}$;取 $D_{1,3}$ 、 $D_{1,4}$ 中的最大值作为 $D_{1,34}$ 。

2. 实现步骤

① 获得所有样品特征。

② 输入阈值 T (计算所有样品距离的最大值与最小值,输出,作为阈值的参考)。

③ 将所有样品各分一类,聚类中心数 $\text{centerNum} = \text{样品总数} \text{ patternNum}$ 。 $\text{m_pattern}(i)$ 。
 $\text{category} = i; \text{m_center}(i)$ 。 $\text{feature} = \text{m_pattern}(i)$ 。 feature 。

④ 对所有样品循环:

➤ 计算所有聚类中心间的距离,两类间的距离等于两类间样品的最大距离 maxDis 。

➤ 取所有 maxDis 中的最小值,设 p_i 类和 p_j 类的距离最小,为 minDis 。

若 $\text{minDis} \leq T$, 则将类号较大的类归入类号较小的类,重新排序类号, $\text{centerNum} = \text{centerNum} - 1$ 。

否则 $\text{minDis} > T$, 即所有类间距离均大于阈值,则退出循环,归类完成。

3. 编程代码

```
%%%%%%%%%%
% 函数名称:C_ZuiChangJuLi()
% 参数:m_pattern:样品特征库;patternNum:样品数目
% 返回值:m_pattern:样品特征库
% 函数功能:按照最长距离法对全体样品进行分类
%%%%%%%%%
function [ m_pattern ] = C_ZuiChangJuLi( m_pattern, patternNum )
    disType = DisSelDlgl(); % 获得距离计算类型
    T = InputThreshDlgl( m_pattern, patternNum, disType ); % 获得阈值
    % 初始化,所有样品各分一类
    for i = 1:patternNum
        m_pattern(i).category = i;
    end
    centerNum = patternNum;
```

```

while( true)
    minDis = inf;
    pi = 0;
    pj = 0;
    % 求两类间的最大距离,找出所有类间距离中最小的为 minDis,记录类号 pi,pj
    for i = 1:centerNum - 1
        for j = i + 1:centerNum
            maxDis = -1;
            for m = 1:patternNum - 1
                for n = m + 1:patternNum
                    if( ( m _ pattern( m) . category == i) && ( m _ pattern( n) . category == j) ||
(m _ pattern( m) . category == j) && ( m _ pattern( n) . category == i) )

                        tempDis = GetDistance( m _ pattern( m) , m _ pattern( n) , disType);
                        if( tempDis > maxDis)
                            maxDis = tempDis;
                        end
                    end
                end
            end

            if( maxDis < minDis)
                minDis = maxDis;
                pi = i;
                pj = j;
            end
        end
    end

    if( minDis < T) % 距离小于阈值,合并 pi、pj 类
        for i = 1:patternNum
            if( m _ pattern( i) . category == pj)
                m _ pattern( i) . category = pi;
            elseif( m _ pattern( i) . category > pj)
                m _ pattern( i) . category = m _ pattern( i) . category - 1;
            end
        end
        centerNum = centerNum - 1;
    else
        break;
    end
end
end

```

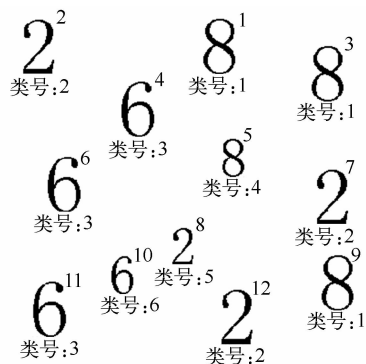
4. 效果图

在最长距离法中,只有两类中的所有样品间的距离都小于阈值,才会被合并成一个类,聚

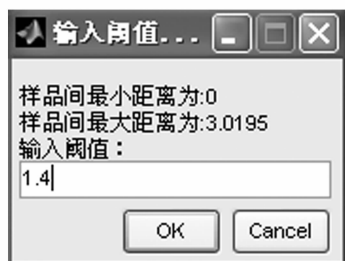
类效果如图 9-6(a),(b),(c),(d)所示,对相同的数据,要想获得同样的聚类效果,最长距离法的阈值要比最短距离法的阈值小一些。如图 9-6(e),(f)所示为最长距离法对图形数据聚类



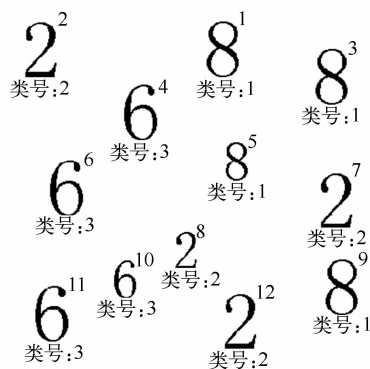
(a) 输入阈值



(b) 阈值过小效果



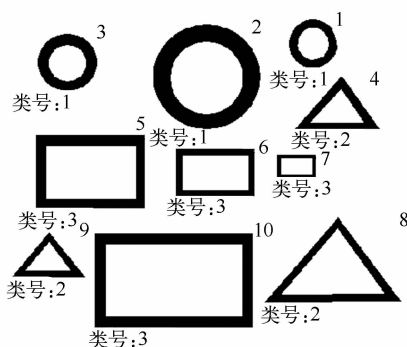
(c) 输入阈值



(d) 阈值合适,聚类正确



(e) 选择夹角余弦距离模式



(f) 聚类效果

图 9-6 最长距离法聚类效果图

9.3.3 中间距离法

1. 理论基础

最短距离法和最长距离法采用的是分类距离的两个极端。中间距离法则介于两者之间,若 ω_j 类由 ω_m, ω_n 两类合并而成,则 ω_i, ω_j 两类的中间距离定义为

$$D_{i,j} = D_{i,mn} = \left(\frac{1}{2}D_{i,m}^2 + \frac{1}{2}D_{i,n}^2 - \frac{1}{4}D_{m,n}^2 \right)^{\frac{1}{2}} \quad (9-4)$$

例如,计算距离 $D_{1,34}$,先计算 ω_1 类中各个样品到 ω_3 类中各个样品的距离 $D_{1,3}$,然后计算 ω_1 类中各个样品到 ω_4 类中各个样品的距离 $D_{1,4}$,及 ω_3 类和 ω_4 类的距离 D_{34} ,按照下式计算 $D_{1,34}$:

$$D_{1,34} = \left(\frac{1}{2}D_{1,3}^2 + \frac{1}{2}D_{1,4}^2 - \frac{1}{4}D_{3,4}^2 \right)^{\frac{1}{2}}$$

2. 实现步骤

- ① 获得所有样品特征。
- ② 输入阈值 T (计算所有样品距离的最大值与最小值,输出,作为阈值的参考)。
- ③ 将所有样品各分一类,聚类中心数 $\text{centerNum} = \text{样品总数}$ patternNum 。 $\text{m_pattern}(i)$.
 $\text{category} = i$; $\text{m_center}(i)$. $\text{feature} = \text{m_pattern}(i)$. feature 。
- ④ 建立距离矩阵 centerDistance ,记录各类间的距离,初始值为各样品间的距离。
- ⑤ 对所有样品循环:
 - 找到 centerDistance 中的最小值 $t_d = \text{centerDistance}(t_i, t_j)$,即 t_i 类和 t_j 类距离最小。
 - 若 $t_d < T$,则将所有 t_j 类成员归入 t_i 类; $\text{centerNum} = \text{centerNum} - 1$;重新顺序排列类号;根据式(9-4)重新计算距离矩阵 **centerDistance**,否则($t_d \geq T$)终止循环,分类结束。

3. 编程代码

```
%%%%%%%%%%
% 函数名称:C_ZhongJianJuLi()
% 参数:m_pattern:样品特征库;patternNum:样品数目
% 返回值:m_pattern:样品特征库
% 函数功能:按照中间距离法对全体样品进行分类
%%%%%%%%%
function [ m_pattern ] = C_ZhongJianJuLi( m_pattern,patternNum )
    disType = DisSelDlg();% 获得距离计算类型
    T = InputThreshDlg( m_pattern,patternNum,disType );% 获得阈值
    % 初始化,所有样品各分一类
    for i = 1:patternNum
        m_pattern(i).category = i;
    end
    centerNum = patternNum;
```



```

if(j < tj)

    tempDistance(i,j) = sqrt( centerDistance( ti,j) * centerDistance( ti,j)/
    2 + centerDistance( j,tj) * centerDistance( j,tj)/2 - centerDistance( ti,
    tj) * centerDistance( ti,tj)/4);

else

    tempDistance(i,j) = sqrt( centerDistance( ti,j + 1) * centerDistance( ti,
    j + 1)/2 + centerDistance( tj,j + 1) * centerDistance( tj,j + 1)/2 -
    centerDistance( ti,tj) * centerDistance( ti,tj)/4);

end
elseif( (i > ti) && (i < tj))
    if(j < tj)
        tempDistance(i,j) = centerDistance(i,j);
    else
        tempDistance(i,j) = centerDistance(i,j + 1);
    end
else
    tempDistance(i,j) = centerDistance(i + 1,j + 1);
end
end
end
end

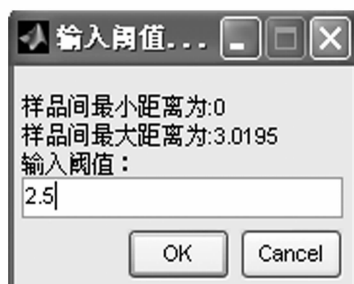
centerDistance = tempDistance;

else
    break;
end
end
end

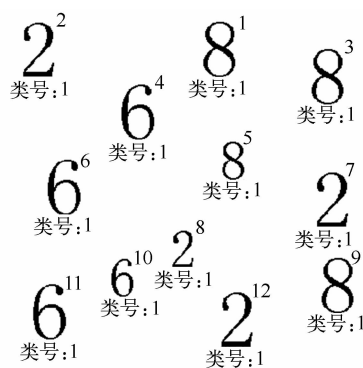
```

4. 效果图

中间距离法聚类效果图如图 9-7 所示,样品的右上角标注该样品编号,左下角为该样品所属类号。



(a) 输入阈值

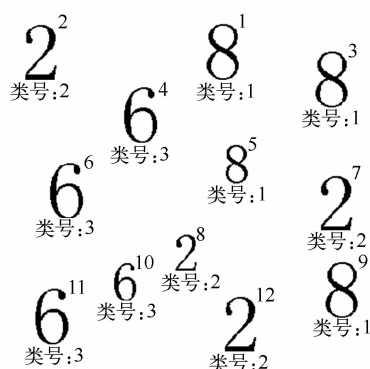


(b) 阈值过大, 不同样品归入同一类

图 9-7 中间距离法聚类效果图



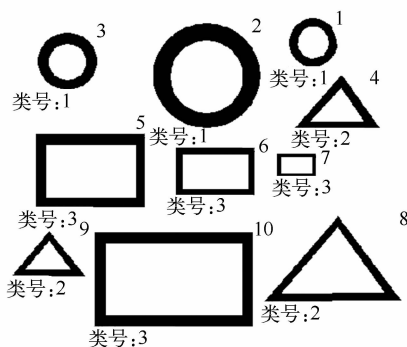
(c) 输入阈值



(d) 阈值合适, 聚类正确



(e) 选择夹角余弦距离模式



(f) 中间距离法应用于图形样品聚类效果

图 9-7 中间距离法聚类效果图(续)

9.3.4 重心法

1. 理论基础

重心法的提出考虑了类中样品个数对类间距离的影响。设 ω_j 类由 ω_m 类和 ω_n 类合并而成, ω_m 类有 N_m 个样品, ω_n 类中有 N_n 个样品。则重心法定义类 ω_i 和类 ω_j 间的距离为

$$D_{i,j} = D_{i,mn} = \left(\frac{N_m}{N_m + N_n} D_{i,m}^2 + \frac{N_n}{N_m + N_n} D_{i,n}^2 - \frac{N_m N_n}{N_m + N_n} D_{m,n}^2 \right)^{\frac{1}{2}} \quad (9-5)$$

例如,要计算 ω_1 到 $\omega_{3,4}$ 的距离 $D_{1,34}$,先计算 ω_1 类中各个样品到 ω_3 类中各个样品的距离 $D_{1,3}$,然后计算 ω_1 类中各个样品到 ω_4 类中各个样品的距离 $D_{1,4}$ 及 ω_3 类和 ω_4 类的距离 $D_{3,4}$ 按照下式计算 $D_{1,34}$:

$$D_{1,34} = \left(\frac{N_3}{N_3 + N_4} D_{1,3}^2 + \frac{N_4}{N_3 + N_4} D_{1,4}^2 - \frac{N_3 N_4}{N_3 + N_4} D_{3,4}^2 \right)^{\frac{1}{2}}$$

2. 实现步骤

① 获得所有样品特征。

② 输入阈值 T (计算所有样品距离的最大值与最小值,输出,作为阈值的参考)。

③ 将所有样品各分一类,聚类中心数 $\text{centerNum} = \text{样品总数 patternNum}$ 。 $\text{m_pattern}(i) . \text{category} = i; \text{m_center}(i) . \text{feature} = \text{m_pattern}(i) . \text{feature}$ 。

④ 建立距离矩阵 **centerDistance**,记录各类间的距离,初始值为各样品间的距离。

⑤ 对所有样品循环:

➤ 找到 **centerDistance** 中的最小值 $t_d = \text{centerDistance}(t_i, t_j)$, 即 t_i 类和 t_j 类距离最小 ($t_j > t_i$)。

➤ 若 $t_d < T$, 则将所有 t_j 类成员归入 t_i 类; $\text{centerNum} = \text{centerNum} - 1$; 重新顺序排列类号; 根据式(9-5)重新计算距离矩阵 **centerDistance**, 否则 ($t_d \geq T$) 终止循环, 分类结束。

3. 编程代码

```
%%%%%%%%%%%%%%
% 函数名称: C_ZhongXin()
% 参数: m_pattern: 样品特征库; patternNum: 样品数目
% 返回值: m_pattern: 样品特征库
% 函数功能: 按照重心法对全体样品进行分类
%%%%%%%%%%%%%
function [ m_pattern ] = C_ZhongXin( m_pattern, patternNum )
    disType = DisSelDlg(); % 获得距离计算类型
    T = InputThreshDlg( m_pattern, patternNum, disType ); % 获得阈值
    % 初始化, 所有样品各分一类
    for i = 1 : patternNum
        m_pattern(i).category = i;
    end
    centerNum = patternNum;
    % 建立类间距离数组, centerdistance(i,j) 表示 i 类和 j 类距离
    centerDistance = zeros( centerNum, centerNum );
    for i = 1 : patternNum - 1
        for j = i + 1 : patternNum
            centerDistance(i,j) = GetDistance( m_pattern(i), m_pattern(j), disType );
        end
    end

    while( true )
        td = inf;
        for i = 1 : centerNum - 1
            for j = i + 1 : centerNum
                if( td > centerDistance(i,j) ) % 找到距离最近的两类: ti, tj, 记录最小距离 td;
                    td = centerDistance(i,j);
                    ti = i;
                    tj = j;
                end
            end
        end
    end
end
```

```

        end
    end
end
numi = 0;
numj = 0;
if(td < T) % 合并类 i,j
    for i = 1:patternNum
        if(m_pattern(i).category == ti)
            numi = numi + 1;
        elseif(m_pattern(i).category == tj)
            m_pattern(i).category = ti;
            numj = numj + 1;
        elseif(m_pattern(i).category > tj)
            m_pattern(i).category = m_pattern(i).category - 1;
        end
    end
    centerNum = centerNum - 1;
    tempDistance = centerDistance; % 临时类间距离矩阵

    for i = 1:centerNum - 1 % 重新计算合并后的类到其他各类的新距离
        for j = i + 1:centerNum
            if(i < ti)
                if(j == ti)

                    tempDistance(i,j) = sqrt(centerDistance(i,ti) * centerDistance(i,ti)
                        * numi/(numi + numj) + centerDistance(i,tj) * centerDistance(i,tj)
                        * numj/(numi + numj) - centerDistance(ti,tj) * centerDistance(ti,tj)
                        * numi * numj/(numi + numj));
                elseif(j > = tj)
                    tempDistance(i,j) = centerDistance(i,j + 1);
                else
                    tempDistance(i,j) = centerDistance(i,j);
                end
            elseif(i == ti)
                if(j < tj)

                    tempDistance(i,j) = sqrt(centerDistance(ti,j) * centerDistance(ti,j)
                        * numi/(numi + numj) + centerDistance(j,tj) * centerDistance(j,tj)
                        * numj/(numi + numj) - centerDistance(ti,tj) * centerDistance(ti,tj)
                        * numi * numj/(numi + numj));
                else

                    tempDistance(i,j) = sqrt(centerDistance(ti,j + 1) * centerDistance(ti,

```

```

        j + 1) * numi / (numi + numj) + centerDistance(tj, j + 1) *
        centerDistance(tj, j + 1) * numj / (numi + numj) - centerDistance(ti, tj)
        * centerDistance(ti, tj) * numi * numj / (numi + numj));
    end
elseif((i > ti) && (i < tj))
    if(j < tj)
        tempDistance(i, j) = centerDistance(i, j);
    else
        tempDistance(i, j) = centerDistance(i, j + 1);
    end
else
    tempDistance(i, j) = centerDistance(i + 1, j + 1);
end
end
end

centerDistance = tempDistance;
else
    break;
end
end
end

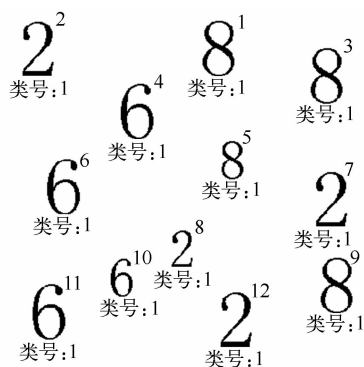
```

4. 效果图

重心法在计算类间的距离时,考虑了样品在多维空间的位置分布对类的重心的影响,计算的结果更能反映类间的不同。重心法聚类效果图如图9-8所示。其中图9-8(e)和(f)是重心法应用于图形聚类的效果。



(a) 阈值过大

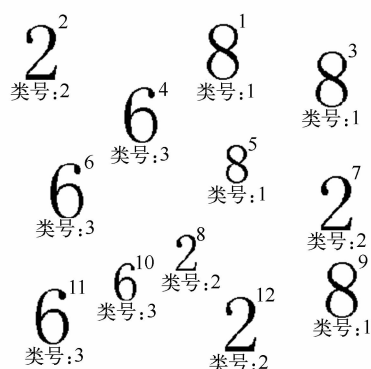


(b) 阈值过大时聚类效果图

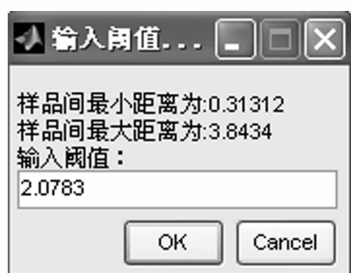
图9-8 重心法聚类效果图



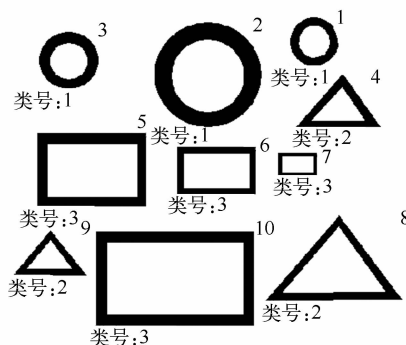
(c) 阈值合适



(d) 正确聚类结果



(e) 阈值合适



(f) 重心法应用于图形聚类效果

图 9-8 重心法聚类效果图(续)

9.3.5 类平均距离法

1. 理论基础

类平均距离的定义也考虑了样品的整体分布特性对其分类的影响。设 ω_j 类由 ω_m 类和 ω_n 类合并而成, ω_m 类有 N_m 个样品, ω_n 类中有 N_n 个样品, ω_i 类中有 N_i 个样品。定义 $D_{i,m}$ 为 ω_i 类和 ω_m 类间的平均距离, $D_{i,n}$ 为 ω_i 类和 ω_n 类间的平均距离。

$$\text{则} \quad D_{i,m} = \left(\frac{1}{N_i N_m} \sum \|X_i - X_m\|^2 \right)^{\frac{1}{2}}$$

$$D_{i,n} = \left(\frac{1}{N_i N_n} \sum \|X_i - X_n\|^2 \right)^{\frac{1}{2}}$$

$$\text{递推可得} \quad D_{i,j} = D_{i,mn} = \left(\frac{N_m}{N_m + N_n} D_{i,m}^2 + \frac{N_n}{N_m + N_n} D_{i,n}^2 \right)^{\frac{1}{2}} \quad (9-6)$$

例如,计算 ω_1 到 $\omega_{3,4}$ 的距离 $D_{1,34}$, 先计算 ω_1 类中各个样品到 ω_3 类中各个样品的平均距离

为 $D_{1,3}$; 然后计算 ω_1 类中各个样品到 ω_4 类中各个样品的平均距离为 $D_{1,4}$ 。

$$D_{1,34} = \left(\frac{N_3}{N_3 + N_4} D_{1,3}^2 + \frac{N_4}{N_3 + N_4} D_{1,4}^2 \right)^{\frac{1}{2}}$$

2. 实现步骤

- ① 获得所有样品特征。
- ② 输入阈值 T (计算所有样品距离的最大值与最小值, 输出, 作为阈值的参考)。
- ③ 将所有样品各分一类, 聚类中心数 $\text{centerNum} = \text{样品总数}$ $\text{patternNum}; \text{m_pattern}(i).$
 $\text{category} = i; \text{m_center}(i). \text{feature} = \text{m_pattern}(i). \text{feature}。$
- ④ 建立距离数组 **centerDistance**, 记录各类间的距离, 初始值为各样品间的距离。
- ⑤ 对所有样品循环:
 - 找到 **centerDistance** 中的最小值 $t_d = \text{centerDistance}(t_i, t_j)$, 即 t_i 类和 t_j 类距离最小 ($t_j > t_i$)。
 - 若 $t_d < T$, 则将所有 t_j 类成员归入 t_i 类; $\text{centerNum} = \text{centerNum} - 1$; 重新顺序排列类号; 根据式(9-6)重新计算距离数组 **centerDistance**, 否则 ($t_d \geq T$) 终止循环, 分类结束。

3. 编程代码

```
%%%%%%%%%%%%%%
% 函数名称: C_LeiPingJunJuLi()
% 参数: m_pattern: 样品特征库; patternNum: 样品数目
% 返回值: m_pattern: 样品特征库
% 函数功能: 按照类平均距离法对全体样品进行分类
%%%%%%%%%%%%%
function [ m_pattern ] = C_LeiPingJunJuLi( m_pattern, patternNum )
    disType = DisSelDlg(); % 获得距离计算类型
    T = InputThreshDlg( m_pattern, patternNum, disType ); % 获得阈值
    % 初始化, 所有样品各分一类
    for i = 1 : patternNum
        m_pattern(i).category = i;
    end
    centerNum = patternNum;
    % 建立类间距离数组, centerdistance(i,j) 表示 i 类和 j 类距离
    centerDistance = zeros( centerNum, centerNum );
    for i = 1 : patternNum - 1
        for j = i + 1 : patternNum
            centerDistance(i,j) = GetDistance( m_pattern(i), m_pattern(j), disType );
        end
    end
    while( true )
        td = inf;
```



```

for i = 1:centerNum - 1
    for j = i + 1:centerNum
        if (td > centerDistance(i,j)) % 找到距离最近的两类:ti,tj,记录最小距离 td;
            td = centerDistance(i,j);
            ti = i;
            tj = j;
        end
    end
end
numi = 0;
numj = 0;

if (td < T) % 合并类 i,j
    for i = 1:patternNum
        if (m_pattern(i).category == ti)
            numi = numi + 1;
        elseif (m_pattern(i).category == tj)
            m_pattern(i).category = ti;
            numj = numj + 1;
        elseif (m_pattern(i).category > tj)
            m_pattern(i).category = m_pattern(i).category - 1;
        end
    end
    centerNum = centerNum - 1;
    tempDistance = centerDistance; % 临时类间距离矩阵

    for i = 1:centerNum - 1 % 重新计算合并后的类到其他各类的新距离
        for j = i + 1:centerNum
            if (i < ti)
                if (j == ti)

                    tempDistance(i,j) = sqrt(centerDistance(i,ti) * centerDistance(i,ti)
                        * numi / (numi + numj) + centerDistance(i,tj) * centerDistance(i,tj)
                        * numj / (numi + numj));
                elseif (j > = tj)
                    tempDistance(i,j) = centerDistance(i,j + 1);
                else
                    tempDistance(i,j) = centerDistance(i,j);
                end
            elseif (i == ti)
                if (j < tj)

                    tempDistance(i,j) = sqrt(centerDistance(ti,j) * centerDistance(ti,j)

```

```

* numi/(numi + numj) + centerDistance(j,tj) * centerDistance(j,tj)
* numj/(numi + numj));
else

tempDistance(i,j) = sqrt( centerDistance(ti,j + 1) * centerDistance(ti,
j + 1) * numi/(numi + numj) + centerDistance(tj,j + 1) *
centerDistance(tj,j + 1) * numj/(numi + numj));

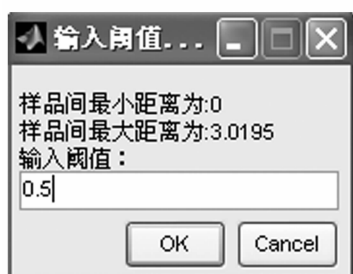
end
elseif((i > ti) && (i < tj))
if(j < tj)
tempDistance(i,j) = centerDistance(i,j);
else
tempDistance(i,j) = centerDistance(i,j + 1);
end
else
tempDistance(i,j) = centerDistance(i + 1,j + 1);
end
end
end

centerDistance = tempDistance;
else
break;
end
end
end

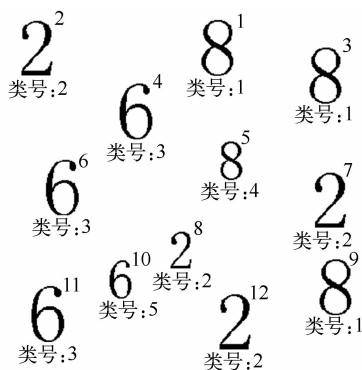
```

4. 效果图

类平均距离法也是应用比较广泛的一种类间距离度量准则。阈值的选取反映了类间的平均距离。由于聚类的标准不同,所以,相同的阈值,不同的聚类方法可能得到不同的聚类结果。类平均距离法聚类效果图如图 9-9 所示。



(a) 阈值过小



(b) 聚类效果

图 9-9 类平均距离法聚类效果图

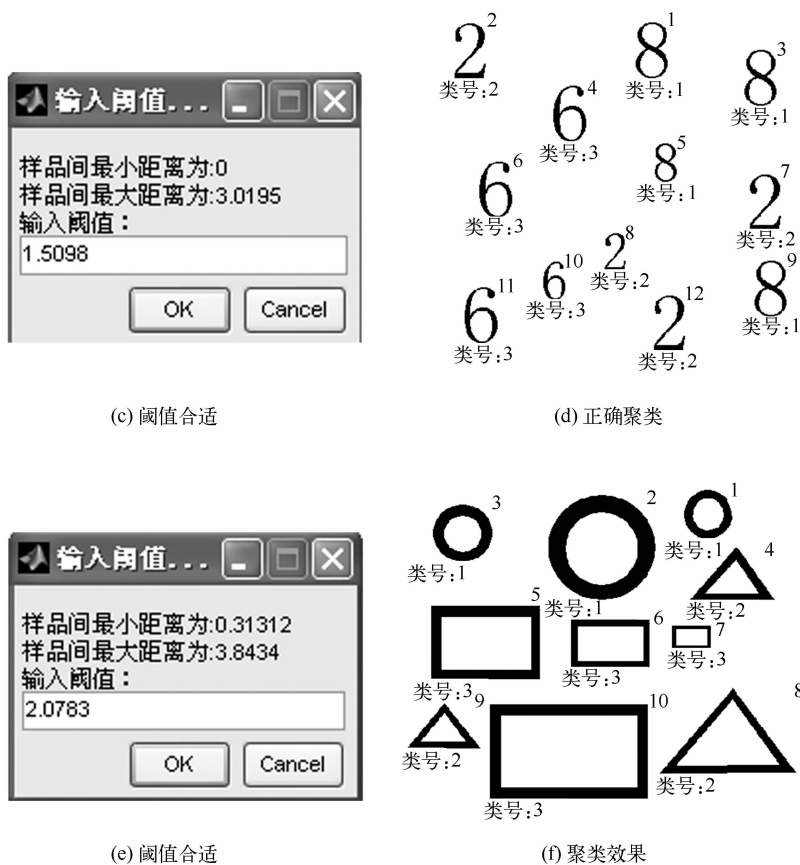


图 9-9 类平均距离法聚类效果图(续)

9.4 动态聚类算法

动态聚类算法选择若干样品作为聚类中心,再按照某种聚类准则,如最小距离准则,将其余样品归入最近的中心,得到初始分类。然后判断初始分类是否合理,若不合理则按照特定规则重新修改不合理的分类,如此反复迭代,直到分类合理。

9.4.1 K 均值算法

1. 理论基础

K 均值算法能够使聚类域中所有样品到聚类中心距离的平方和最小。其原理为:先取 k 个初始距离中心,计算每个样品到这 k 个中心的距离,找出最小距离把样品归入最近的聚类中心,如图 9-10(a)所示,修改中心点的值为本类所有样品的均值,再计算各个样品到 k 个中心的距离,重新归类、修改新的中心点,如图 9-10(b)所示。直到新的距离中心等于上一次的中心点时结束。此算法的结果受到聚类中心的个数以及初始聚类中心的选择影响,也受到样品几何性质及排列次序影响。如果样品的几何特性表明它们能形成几个相距较远的小块孤立区域,则算法多能收敛。

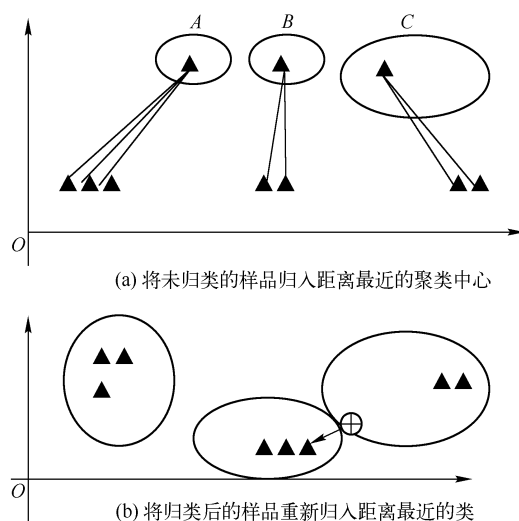


图 9-10 K 均值算法示意图

2. 实现步骤

① 通过对话框读取需要分类数目 $centerNum$, 和最大迭代次数 $iterNum$ 。

② 随机取 $centerNum$ 个样品作为聚类中心。 $m_center(i).feature = m_pattern(i).feature$, $m_center(i).index = i$; $m_pattern(i).category = i$; $i = (1 \sim centerNum)$, 其余样品中心号为 -1, 样品到本类中心的距离为 \max (\max 为无穷大)。

③ 假设前三个样品分别属于每一类, 需要分三类 A、B、C, 参见图 9-10(a), 计算其余样品到这三个类的距离, 将它们归为距离最近的类, 至此, 所有的样品都归类完毕。计算各个类中心所有样品特征值的平均值作为该聚类中心的特征值。

④ 如图 9-10(b) 所示, 对每一类中的各个样品, 计算它到其他类中心的距离, 如果它到某一类中心的距离小于它到自身类中心的距离, 需要对该样品重新分类, 将它归属到距离中心近的类, 循环重复所有的样品, 直至不再有样品类号发生变化。

3. 编程代码

本算法编程采用了两种方式: 一种是按实现步骤编写的 K 均值算法聚类; 另一种是使用 Matlab 工具箱中的 K 均值算法聚类(通过 $Kmeans()$ 函数实现)。

```
%%%%%%%%%%
% 函数名称: C_KJunZhi()
% 参数: m_pattern: 样品特征库; patternNum: 样品数目
% 返回值: m_pattern: 样品特征库
% 函数功能: 按照 K 均值法对全体样品进行分类
%%%%%%%%%
function [ m_pattern ] = C_KJunZhi( m_pattern, patternNum )
    disType = DisSelDlg(); % 获得距离计算类型
    [ centerNum iterNum ] = InputClassDlg(); % 获得类中心数和最大迭代次数
```

```

for i = 1:patternNum
    m_pattern(i).distance = inf;
    m_pattern(i).category = -1;
end
randPattern = randperm(patternNum);
for i = 1:centerNum% 初始化,随机分配 centerNum 个粒子为一类
    m_pattern(randPattern(i)).category = i;
    m_pattern(randPattern(i)).distance = 0;
    m_center(i).feature = m_pattern(randPattern(i)).feature;
    m_center(i).index = i;
    m_center(i).patternNum = 1;
end
counter = 0;% 记录当前已经循环的次数
change = 1;
while(counter < iterNum && change ~ = 0)
    counter = counter + 1;
    change = 0;
    for i = 1:patternNum% 对所有样品重新归类
        % 计算第 i 个模式到各个聚类中心的最小距离
        index = -1;
        distance = inf;
        for j = 1:centerNum
            tempDis = GetDistance(m_pattern(i),m_center(j),disType);
            if(distance > tempDis)
                distance = tempDis;
                index = j;
            end
        end
        % 比较原中心号与新中心号
        % 相同:更新距离。
        % 不同:1,新距离小,则归入新中心,更新距离,重新计算前后两个聚类中心模式
        % 2,新距离大于原距离,不处理;

        if(m_pattern(i).category == index)% 属于原类
            m_pattern(i).distance = distance;
        else% 不属于原类
            oldIndex = m_pattern(i).category;% 记录原类号
            m_pattern(i).category = index;% 归入新类
            m_pattern(i).distance = distance;
            if(oldIndex ~ = -1)
                m_center(oldIndex) = CalCenter(m_center(oldIndex),m_pattern,patternNum);
            end
            m_center(index) = CalCenter(m_center(index),m_pattern,patternNum);
            change = 1;
        end
    end
end

```

```

end
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 函数名称:C_KJunZhi2( )
% 参数:m_pattern:样品特征库;patternNum:样品数目
% 返回值:m_pattern:样品特征库
% 函数功能:按照 K 均值法对全体样品进行分类( Matlab 工具箱版本)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ m_pattern ] = C_KJunZhi2( m_pattern,patternNum )
    str1 = '类中心数:';
    T = inputdlg( str1,'输入对话框' );
    centerNum = str2num( T{1,1} );% 获得类中心数
    global Nwidth;
    X = zeros( Nwidth * Nwidth,patternNum );
    for i = 1:patternNum
        X(:,i) = m_pattern(i).feature(:);
    end
    try
        IDX = kmeans( X', centerNum );% K 均值算法( matlab 工具箱函数)
    catch
        msgbox('本实例无法用 Matlab 工具箱 K 均值算法聚类,请尝试另一种 K 均值算法','modal');
        for i = 1:patternNum
            m_pattern(i).category = 0;
        end
        return;
    end
    for i = 1:patternNum
        m_pattern(i).category = IDX(i);
    end
end

```

4. 效果图

K 均值算法是一种动态聚类算法。用户只需输入计算距离的类型,如图 9-11(a)所示,输入聚类中心数目和迭代次数,效果如图 9-11(b)和(c)所示。利用 Matlab 工具箱中的 K 均值算法聚类只需输入聚类中心数目,其效果如图 9-12 所示。



(a) 选择夹角余弦距离模式



(b) 输入类中心数和最大迭代次数

图 9-11 K 均值算法聚类效果图

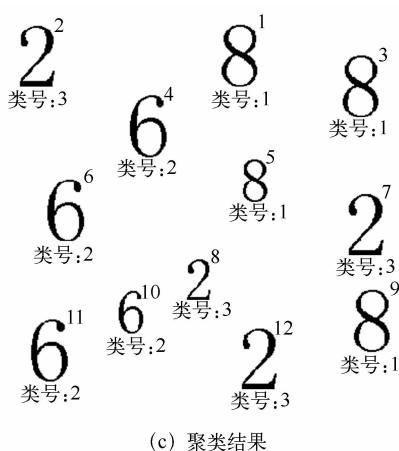


图 9-11 K 均值算法聚类效果图(续)

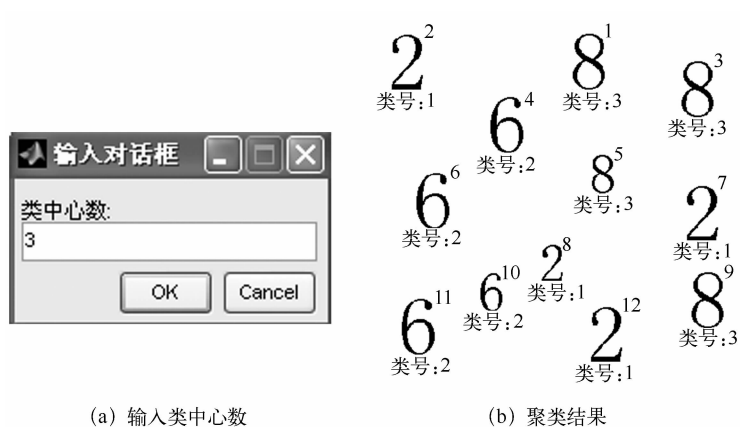


图 9-12 Matlab 工具箱中 K 均值算法聚类效果图

9.4.2 迭代自组织的数据分析算法 (ISODATA)

1. 理论基础

迭代自组织的数据分析算法 (Iterative Self-organizing Data Analysis Techniques Algorithm) 也称 ISODATA 算法。此算法与 K 均值算法有相似之处,即聚类中心也是通过样品均值的迭代运算来决定的。但 ISODATA 加入了一些试探性的步骤,能吸取中间结果所得到的经验,在迭代过程中可以将一类一分为二,也可以将两类合并,即“自组织”。这种算法具有启发性。

2. 实现步骤

- ① 获得所有样品特征。

② 输入阈值 T , 方差 $equation$, 类中心数目 $centerNum$, 最大迭代次数 $iterNum$ (计算所有样品距离的最大值与最小值, 以及方差的最小最大值, 输出, 作为阈值的参考)。

③ 任意选取 $precenterNum$ 个 (不妨取前 $centerNum$ 个) 样品作为聚类中心 $m_center(i)$ 。

④ 求各个样品到所有聚类中心的距离, 将所有样品归入最近的类中心 $m_center(i)$ 。

⑤ 修正各聚类中心的值。

⑥ 计算各聚类域中诸样品到聚类中心间的平均距离。

⑦ 计算所有聚类域样品平均距离的总平均距离。

⑧ 判断分裂、合并及迭代等步骤:

➤ 若迭代次数已达到 $iterNum$, 置 $equation = 0$, 跳到第⑩步, 运算结束。

➤ 若 $precenterNum > 2 \times centerNum$, 或者进行了偶数次迭代并且 $precenterNum > centerNum/2$, 则进入第⑨步, 合并处理。否则, 转第⑩步分裂处理。

⑨ 合并操作, 计算全部聚类中心的距离, 设 $t_i, t_j (t_i < t_j)$ 距离最近, 设最小距离 t_d 。若 $t_d < T$ (阈值), 则将 t_j 类并入 t_i 类。 $precenterNum - 1$ 。计算和并后的新中心。

⑩ 分裂操作, 求所有聚类中心的标准差向量 σ_i , $\sigma_i = \sqrt{\frac{1}{N_i} \sum_{X \in \omega_i} (X - \bar{X}^{(\omega_i)})^2}$, $i = 1, 2, \dots, precenterNum$, N_i 为 ω_i 类中样品个数。找到所有中心标准差中的最大值, 设第 t_i 类的第 t_j 位标准差最大, 最大值为 $mequation$ 。

若 $mequation > equation$, 则 $precenterNum++$, 新中心特征值等于 $m_center(t_i)$ 的特征值, 只是第 t_j 位需要调整,

$$m_center(t_i).feature(t_j) = m_center(t_i).feature(t_j) + a \times mequation,$$

$$m_center(precenterNum - 1).feature(t_j) = m_center(t_i).feature(t_j) - a \times mequation;$$

其中 $a = (0, 1)$, 取 $a = 0.5$ 。

⑪ 如果是最后一次迭代运算 (即第 $iterNum$ 次迭代) 则结束循环。否则循环继续第④步, 迭代次数加 1。

3. 编程代码

```
%%%%%%%%%%
```

```
% 函数名称: C_ISODATA()
```

```
% 参数: m_pattern: 样品特征库; patternNum: 样品数目
```

```
% 返回值: m_pattern: 样品特征库
```

```
% 函数功能: 按照 ISODATA 法对全体样品进行分类
```

```
%%%%%%%%%%
```

```
function [ m_pattern ] = C_ISODATA( m_pattern, patternNum )
```

```
    disType = DisSelDlg(); % 获得距离计算类型
```

```
    [ T, equation, centerNum, iterNum ] = InputIsodataDlg( m_pattern, patternNum, disType );
```

```
    precenterNum = centerNum;
```

```
    for i = 1:precenterNum % 初始化, 前 centerNum 个模板各自分为一类
```

```
        m_pattern(i).category = i;
```

```
        m_center(i).feature = m_pattern(i).feature;
```



```

        m_center(i).index = i;
        m_center(i).patternNum = 1;
    end
    counter = 0; % 循环次数
    while( counter < iterNum)
        counter = counter + 1;
        change = 0;
        for i = 1:patternNum % 对所有样品重新归类
            % 计算第 i 个模式到各个聚类中心的最小距离
            index = -1;
            td = inf;
            for j = 1:precenterNum
                tempDis = GetDistance( m_pattern(i), m_center(j), disType);
                if( td > tempDis)
                    td = tempDis;
                    index = j;
                end
            end
            m_pattern(i).category = m_center(index).index;
        end
        % 修正各中心
        for i = 1:precenterNum
            m_center(i) = CalCenter( m_center(i), m_pattern, patternNum);
        end
        for i = 1:precenterNum
            if( m_center(i).patternNum == 0)
                for j = i:precenterNum - 1
                    m_center(j) = m_center(j + 1);
                end
                precenterNum = precenterNum - 1;
            end
        end
        aveDistance = zeros( centerNum); % 计算各类距中心平均距离
        allAveDis = 0; % 全部样本平均距离
        for i = 1:precenterNum
            num = 0; % 类中成员个数
            dis = 0;
            for j = 1:patternNum
                if( m_pattern(j).category == m_center(i).index)
                    num = num + 1;
                    dis = dis + GetDistance( m_pattern(j), m_center(i), disType);
                end
            end
            end
        end
    end
end

```

```

    allAveDis = allAveDis + dis;
    aveDistance(i) = dis/num;
end
allAveDis = allAveDis/patternNum;

if ((precenterNum >= 2 * centerNum) || ((mod(counter, 2) == 0) && (precenterNum >
centerNum/2))) % 合并
    % 找到距离最近的两个类
    td = inf;
    for i = 1:precenterNum
        for j = i + 1:precenterNum
            tempDis = GetDistance(m_center(i), m_center(j), disType);
            if(td > tempDis)
                td = tempDis;
                ti = i;
                tj = j;
            end
        end
    end
    % 判断是否要合并
    if(td < T) % 合并
        for i = 1:patternNum
            if(m_pattern(i).category == m_center(tj).index)
                m_pattern(i).category = m_center(ti).index;
            elseif(m_pattern(i).category > m_center(tj).index)
                m_pattern(i).category = m_pattern(i).category - 1;
            end
        end
    end
else % 分裂
    global Nwidth;
    % 计算标准差
    for i = 1:precenterNum
        mEquation(i).equ = zeros(Nwidth, Nwidth);
        for j = 1:patternNum
            if(m_pattern(j).category == m_center(i).index)
                mEquation(i).equ = mEquation(i).equ + (m_pattern(j).feature - m_center(i).feature).^2;
            end
        end
        mEquation(i).equ = sqrt(mEquation(i).equ/m_center(i).patternNum);
    end
    % 找最大标准差
    ti = 1;

```

```

tm = 1;
tn = 1;
for i = 1:precenterNum
    for m = 1:Nwidth
        for n = 1:Nwidth
            if( mEquation(i).equ(m,n) > mEquation(ti).equ(tm,tn) )
                ti = i;
                tm = m;
                tn = n;
            end
        end
    end
end
% 判断是否要分裂
if( mEquation(ti).equ(tm,tn) > equation) % 大于阈值
    if( aveDistance(ti) > allAveDis) % 类平均距离大于总平均距离,分裂
        precenterNum = precenterNum + 1;
        for i = 1:precenterNum - 1
            tempCenter(i) = m _ center(i);
        end
        tempCenter(precenterNum).index = precenterNum;
        tempCenter(precenterNum).feature = m _ center(ti).feature;

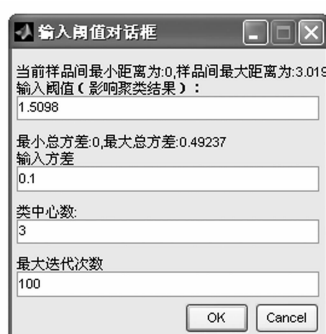
        tempCenter(precenterNum).feature(tm,tn) = tempCenter(precenterNum).
            feature(tm,tn) + 0.5 * mEquation(ti).equ(tm,tn);

        tempCenter(precenterNum - 1).feature(tm,tn) = tempCenter(precenterNum).
            feature(tm,tn) - 0.5 * mEquation(ti).equ(tm,tn);
        m _ center = tempCenter;
    end
end
end
end
end

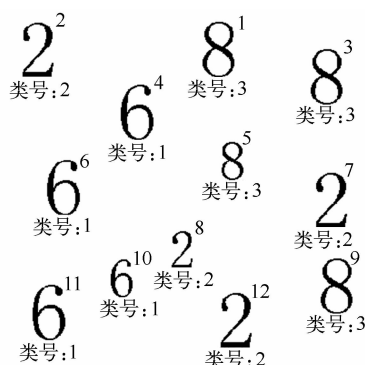
```

4. 效果图

ISODATA 算法具有自组织性,会在计算过程中不断地调整类中心的个数,直到使分类的总的样品方差最小。很显然,当所有样品各分一类的时候,总的样品方差为零,但这样的聚类结果毫无意义。选择不同的距离类型,在用户输入栏的“阈值”一项中会给出参考值,如图 9-13(a)和(c)所示,类中心数目应该小于样品数目。ISODATA 算法聚类效果图如图 9-13 所示。



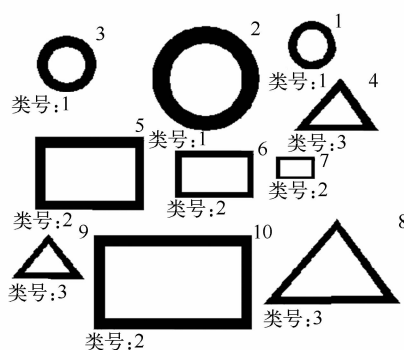
(a) 输入相关信息



(b) 聚类结果



(c) 对不规则符号归类



(d) 聚类结果

图 9-13 ISODATA 算法聚类效果图

9.5 模拟退火聚类算法

9.5.1 模拟退火的基本概念

模拟退火算法(Simulated annealing, SA)最初由 Metropolis 等人于 20 世纪 80 年代初提出,其思想源于物理中固体物质退火过程与一般组合优化问题之间的相似性。模拟退火方法是一种通用的优化算法,目前已广泛应用于最优控制、机器学习、神经网络等优化问题。

1. 物理退火过程

模拟退火算法源于物理中固体物质退火过程,整个过程由以下三部分组成。

(1) 升温过程

升温的目的是增强物体中粒子的热运动,使其偏离平衡位置变为无序状态。当温度足够高时,固体将溶解为液体,从而消除系统原先可能存在的非均匀态,使随后的冷却过程以某一平衡态为起点。升温过程与系统的熵增过程相关,系统能量随温度升高而增大。

(2) 等温过程

在物理学中,对于与周围环境交换热量而温度不变的封闭系统,系统状态的自发变化总是朝向自由能减小的方向进行,当自由能达到最小时,系统达到平衡态。

(3) 冷却过程

与升温过程相反,使物体中粒子的热运动减弱并渐趋有序,系统能量随温度降低而下降,得到低能量的晶体结构。

2. 模拟退火算法的基本原理

模拟退火的基本思想是指将固体加温至充分高,再让其徐徐冷却,加温时,固体内部粒子随温升变为无序状,内能增大,而徐徐冷却时粒子渐趋有序,在每个温度都达到平衡态,最后在常温时达到基态,内能减为最小。

根据 Metropolis 准则,粒子在温度 T 时趋于平衡的几率为 $e^{-\Delta E/(kT)}$,其中 E 为温度 T 时的内能, ΔE 为其改变量, K 为 Boltzmann 常数。用固体退火模拟组合优化问题,将内能 E 模拟为目标函数值 f ,温度 T 演化成控制参数 t ,即得到解组合优化问题的模拟退火算法:由初始解和控制参数初值开始,对当前解重复“产生新解→计算目标函数差→判断是否接受→接受或舍弃”的迭代,并逐步衰减 t 值,算法终止时的当前解即为所得近似最优解,这是蒙特卡罗迭代求解法的一种启发式随机搜索过程。

如果用粒子的能量定义材料的状态, Metropolis 算法用一个简单的数字模型描述了退火过程。假设材料在状态 i 之下的能量为 $E(i)$,那么材料在温度 T 时从状态 i 进入到状态 j 就遵循如下规律:

如果 $E(j) \leq E(i)$,则接收该状态被转换;

如果 $E(j) > E(i)$,则状态转换以如下概率被接收:

$$p = e^{(E(i) - E(j))/(KT)} \quad (9-7)$$

式中, K 为物理学中的常数; T 为材料的温度。

(1) 模拟退火算法的组成

模拟退火算法由解空间、目标函数和初始解三部分组成。

① 解空间:对所有可能解均为可行解的问题定义为可能解的集合,对存在不可行解的问题,或限定解空间为所有可行解的集合,或允许包含不可行解但在目标函数中用惩罚函数(Penalty Function)惩罚以致最终完全排除不可行解。

② 目标函数:对优化目标的量化描述,是解空间到某个数集的一个映射,通常表示为若干优化目标的一个和式,应正确体现问题的整体优化要求且较易计算,当解空间包含不可行解时还应包括罚函数项。

③ 初始解:是算法迭代的起点,试验表明,模拟退火算法是健壮的(Robust),即最终解的求得不十分依赖初始解的选取,从而可任意选取一个初始解。

(2) 模拟退火算法的基本过程

① 初始化,给定初始温度 T_0 及初始解 ω ,计算解对应的目标函数值 $f(\omega)$,在本节中 ω 代表一种聚类划分。

② 模型扰动产生新解 ω' 及对应的目标函数值 $f(\omega')$ 。

③ 计算函数差值 $\Delta f = f(\omega') - f(\omega)$ 。

④ 如果 $\Delta f \leq 0$, 则接受新解作为当前解。

⑤ 如果 $\Delta f > 0$, 则以概率 p 接受新解。

$$p = e^{-(f(\omega') - f(\omega)) / (KT)} \quad (9-8)$$

⑥ 对当前 T 值降温, 对步骤②~⑤迭代 N 次。

⑦ 如果满足终止条件, 输出当前解为最优解, 结束算法, 否则降低温度, 继续迭代。

模拟退火算法流程如图 9-14 所示。算法中包含 1 个内循环和 1 个外循环, 内循环就是在同一温度下的多次扰动产生不同模型状态, 并按照 Metropolis 准则接受新模型, 因此是用模型扰动次数控制的; 外循环包括了温度下降的模拟退火算法的迭代次数的递增和算法停止的条件, 因此基本是用迭代次数控制的。

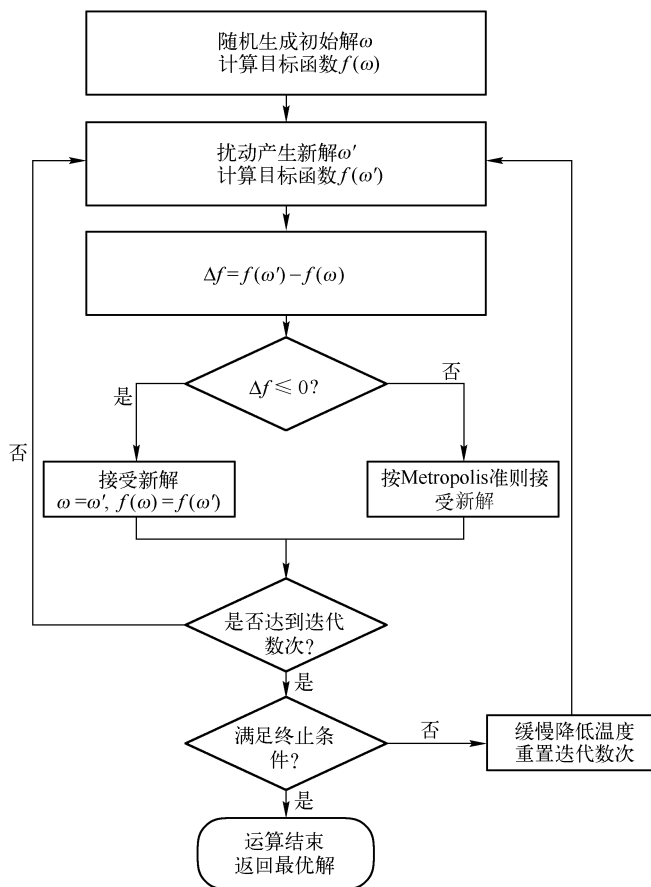


图 9-14 模拟退火算法流程图

3. 退火方式

模拟退火算法中, 退火方式对算法有很大的影响。如果温度下降过慢, 算法的收敛速度会大大降低。如果温度下降过快, 可能会丢失极值点。为了提高模拟退火算法的性能, 许多学者

提出了退火方式,比较有代表性的几种退火方式如下:

$$\textcircled{1} \quad T(t) = \frac{T_0}{\ln(1+t)} \quad (9-9)$$

t 代表图 9-14 中的最外层当前循环次数,其特点是温度下降缓慢,算法收敛速度也较慢。

$$\textcircled{2} \quad T(t) = \frac{T_0}{\ln(1+at)} \quad (9-10)$$

a 为可调参数,可以改善退火曲线的形态。其特点是高温区温度下降较快,低温区是温度下降较慢,即主要在低温区进行寻优。

$$\textcircled{3} \quad T(t) = T_0 \cdot a^t \quad (9-11)$$

a 为可调参数。其特点是温度下降较快,算法收敛速度快。

9.5.2 基于模拟退火思想的改进 K 均值聚类算法

1. K 均值算法的局限性

基本的 K 均值算法目的是找到使目标函数值最小的 K 个划分,算法思想简单,易实现,而且收敛速度较快。如果各个簇之间区别明显,且数据分布稠密,则该算法比较有效,但如果各个簇的形状和大小差别不大,则可能会出现较大的簇分割现象。此外,在 K 均值算法聚类时,最佳聚类结果通常对应于目标函数的极值点,由于目标函数可能存在很多的局部极小值点,这就会导致算法在局部极小值点收敛。因此初始聚类中心的随机选取可能会使解陷入局部最优解,难以获得全局最优解。

该算法主要的局限性主要表现为:

- ① 最终的聚类结果依赖于最初的划分。
- ② 需要事先指定聚类的数目 M 。
- ③ 产生的类大小相关较大,对于“噪声”和孤立点敏感。
- ④ 算法经常陷入局部最优。
- ⑤ 不适合对非凸面形状的簇或差别很小的簇进行聚类。

2. 基于模拟退火思想的改进 K 均值聚类算法

模拟退火算法是一种启发式随机搜索算法,具有并行性和渐近收敛性,已在理论上证明它是一种以概率为 1,收敛于全局最优解的全局优化算法,因此用模拟退火算法对 K 均值聚类算法进行优化,可以改进 K 均值聚类算法的局限性,提高算法性能。

基于模拟退火思想的改进 K 均值聚类算法中,将内能 E 模拟为目标函数值,将基本 K 均值聚类算法的聚类结果作为初始解,初始目标函数值作为初始温度 T_0 ,对当前解重复“产生新解→计算目标函数差→接受或舍弃新解”的迭代过程,并逐步降低 T 值,算法终止时当前解为近似最优解。这种算法开始时以较快的速度找到相对较优的区域,然后进行更精确的搜索,最终找到全局最优解。

3. 几个重要参数的选择

(1) 目标函数

选择当前聚类划分的总类间离散度作为目标函数,如式 9-12 所示。

$$J_{\omega} = \sum_{i=1}^M \sum_{X \in \omega_i} d(X, \overline{X^{(\omega_i)}}) \quad (9-12)$$

式中, X 为样本向量; ω 为聚类划分; $\overline{X^{(\omega_i)}}$ 为第 i 个聚类的中心; $d(X, \overline{X^{(\omega_i)}})$ 为样品到对应聚类中心距离; 聚类准则函数 J_{ω} 即为各类样本到对应聚类中心距离的总和。

(2) 初始温度

一般情况下,为了使最初产生的新解被接受,在算法开始时就应达到准平衡。因此选取基本 K 均值聚类算法的聚类结果作为初始解,初始温度 $T_0 = J_{\omega}$ 。

(3) 扰动方法

模拟退火算法中的新解的产生是对当前解进行扰动得到的。本算法采用一种随机扰动方法,即随机改变一个聚类样品的当前所属类别,从而产生一种新的聚类划分,从而使算法有可能跳出局部极小值。

(4) 退火方式

本算法采用式(9-11)描述的退火方式,其中 a 为退火速度,控制温度下降的快慢,取 $a = 0.99$ 。

4. 算法流程

基于模拟退火思想的改进 K 均值聚类算法流程如图 9-15 所示。

5. 实现步骤

实现步骤如下所述。

① 对样品进行 K 均值聚类,将聚类划分结果作为初始解 ω ,根据式(9-12)计算目标函数值 J_{ω} 。

② 初始化温度 T_0 ,令 $T_0 = J_{\omega}$ 。初始化退火速度 a 和最大退火次数。

③ 对于某一温度 t ,在步骤④~⑦进行迭代,直到达到最大迭代次数跳到步骤⑧。

④ 随机扰动产生新的聚类划分 ω' ,即随机改变一个聚类样品的当前所属类别,计算新的目标函数值 $J_{\omega'}$ 。

⑤ 判断新的目标函数值 $J_{\omega'}$ 是否为最优目标函数值,是则保存聚类划分 ω' 为最优聚类划分、 $J_{\omega'}$ 为最优目标函数值;否则跳到下一步。

⑥ 计算新的目标函数值与当前目标函数值的差 ΔJ 。

⑦ 判断 ΔJ 是否小于 0:

➤ 若 $\Delta J < 0$,则接受新解,即将新解作为当前解。

➤ 若 $\Delta J \geq 0$,则根据 Metropolis 准则,以概率 $p(p = e^{\Delta J/Kt})$ 接受新解。 K 为常数, t 为当前温度。

⑧ 判断是否达到最大退火次数,是则结束算法,输出最优聚类划分;否则根据退火公式(9-11)对温度 t 进行退火,返回步骤③继续迭代。

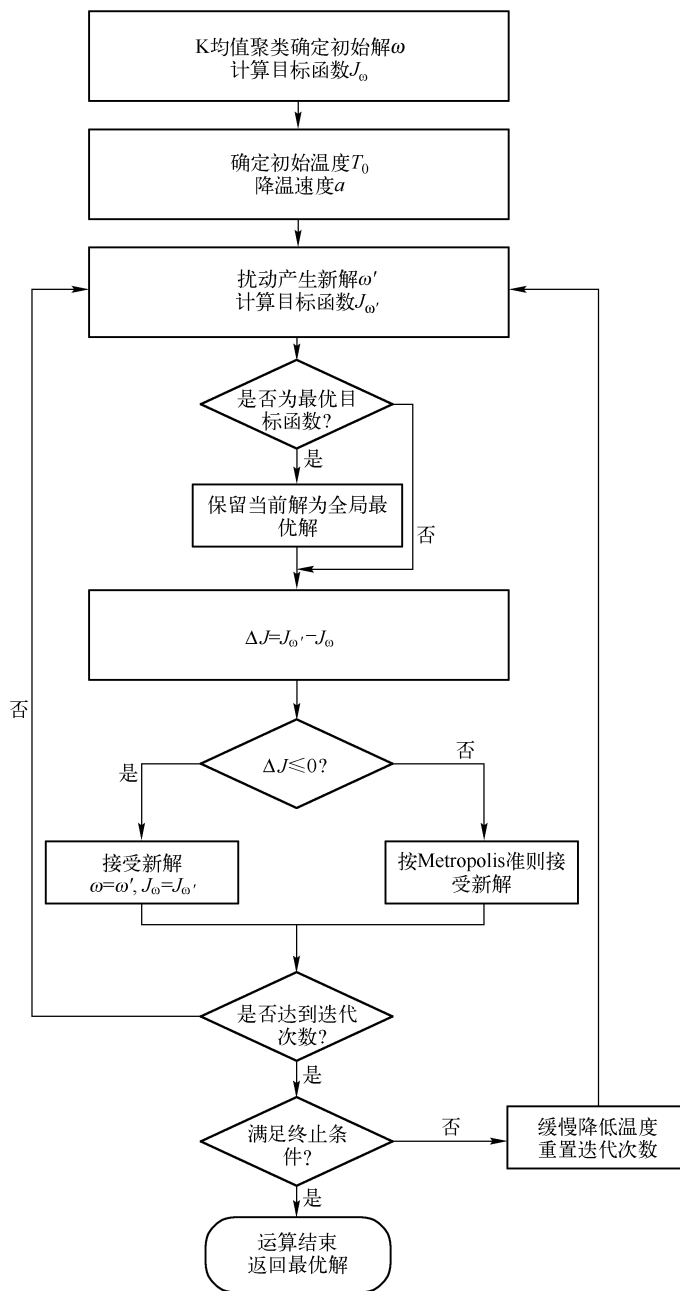


图 9-15 基于模拟退火思想的改进 K 均值聚类算法流程图

6. 编程代码

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 函数名称:C_MoNiTuiHuo()
% 参数:m_pattern:样品特征库;patternNum:样品数目
% 返回值:m_pattern:样品特征库

```

```

% 函数功能          按照基于模拟退火的 K 均值算法对全体样品进行分类
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function ( m _ pattern ) = C _ MoNiTuiHuo( m _ pattern , patternNum )
    disType = DisSelDlg( ) ; % 获得距离计算类型
    ( centerNum iterNum Tn Ts ) = InputTuiHuoDlg( ) ; % 获得类中心数和最大迭代次数,最大退火
    次数,退火速度
    for i = 1 : patternNum
        m _ pattern( i ) . distance = inf ;
        m _ pattern( i ) . category = - 1 ;
    end
    randPattern = randperm( patternNum ) ;
    for i = 1 : centerNum % 初始化,随机分配 centerNum 个粒子为一类
        m _ pattern( randPattern( i ) ) . category = i ;
        m _ pattern( randPattern( i ) ) . distance = 0 ;
        m _ center( i ) . feature = m _ pattern( randPattern( i ) ) . feature ;
        m _ center( i ) . index = i ;
        m _ center( i ) . patternNum = 1 ;
    end
    counter = 0 ; % 记录当前已经循环的次数
    change = 1 ;
    while( counter < iterNum && change ~ = 0 )
        counter = counter + 1 ;
        change = 0 ;
        for i = 1 : patternNum % 对所有样品重新归类
            % 计算第 i 个模式到各个聚类中心的最小距离
            index = - 1 ;
            distance = inf ;
            for j = 1 : centerNum
                tempDis = GetDistance( m _ pattern( i ) , m _ center( j ) , disType ) ;
                if( distance > tempDis )
                    distance = tempDis ;
                    index = j ;
                end
            end
            % 比较原中心号与新中心号
            % 相同:更新距离
            % 不同:1,新距离小,则归入新中心,更新距离,重新计算前后两个聚类中心模式
            % 2,新距离大于原距离,不处理

            if( m _ pattern( i ) . category == index ) % 属于原类
                m _ pattern( i ) . distance = distance ;
            else % 不属于原类
                oldIndex = m _ pattern( i ) . category ; % 记录原类号
                m _ pattern( i ) . category = index ; % 归入新类
                m _ pattern( i ) . distance = distance ;
            end
        end
    end
end

```

```

        if(oldIndex ~= -1)
            m_center(oldIndex) = CalCenter(m_center(oldIndex), m_pattern, patternNum);
        end
        m_center(index) = CalCenter(m_center(index), m_pattern, patternNum);
        change = 1;
    end
end
end
% 计算目标函数
AimFunc = 0;
for j = 1:patternNum
    AimFunc = AimFunc + GetDistance(m_pattern(j), m_center(m_pattern(j).category),
        disType);
end
AimOld = AimFunc;
oldCenter = m_center;
oldPattern = m_pattern;
Tc = 1; % 当前退火次数
bestAim = AimOld; % 最优目标函数
bestPattern = m_pattern;
MarkovLength = 1000;
Tb = 0; % 最优目标函数首次出现的退火次数
T = AimFunc; % 初始化温度参数
str = ('K 均值算法, 最优目标函数值: ' num2str(bestAim));
disp(str);
while(Tc <= Tn && bestAim > 0.1)
    for inner = 1:MarkovLength
        % 产生随机扰动
        p = fix(rand * patternNum + 1);
        t = fix(rand * (centerNum - 1) + 1);
        if(m_pattern(p).category + t > centerNum)
            m_pattern(p).category = m_pattern(p).category + t - centerNum;
        else
            m_pattern(p).category = m_pattern(p).category + t;
        end
        % 重新计算聚类中心
        for i = 1:centerNum
            m_center(i) = CalCenter(m_center(i), m_pattern, patternNum);
        end
        AimFunc = 0;
        % 计算目标函数
        for j = 1:patternNum
            AimFunc = AimFunc + GetDistance(m_pattern(j), m_center(m_pattern(j).
                category), disType);
        end
    end
end

```

```

e = AimFunc - AimOld;
% 记录最优聚类
if( AimFunc < bestAim)
    bestAim = AimFunc;
    bestPattern = m _ pattern;
    Tb = Tc;
end
if( bestAim == 0)
    break;
end
% 判断是否接受新解
if( e < 0)
    AimOld = AimFunc;
else
    k = exp( - e/T );
    if( rand < exp( - e/T ) )
        AimOld = AimFunc;
    else
        m _ pattern = oldPattern;
        m _ center = oldCenter;
    end
end
end

end
T = T * Ts;
if( T == 0)
    break;
end;
Tc = Tc + 1;
if( Tc - Tb > Tn/2 ) % 连续 Tn/2 次退火无改变, 结束退火
    break;
end
str = ('已退火 ' num2str( Tc - 1 ) '次;' 最优目标函数值: ' num2str( bestAim ) );
disp( str );
m _ pattern = bestPattern;
end
m _ pattern = bestPattern;
str = ('当前最优解出现时, 已退火次数为: ' num2str( Tb ) );
msgbox( str, 'modal' );

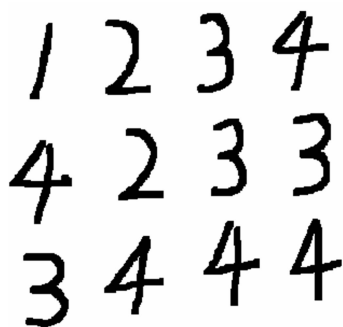
```

7. 效果图

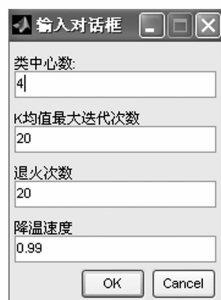
基于模拟退火思想的改进 K 均值聚类算法采用了 Metropolis 准则, 故成为全局寻优算法。Metropolis 准则及算法的优点是: 中间解以一定的接受概率跳出局部极小, 避免落入局部极小

点的可能,然后在退火温度的控制下最终找到最优解。

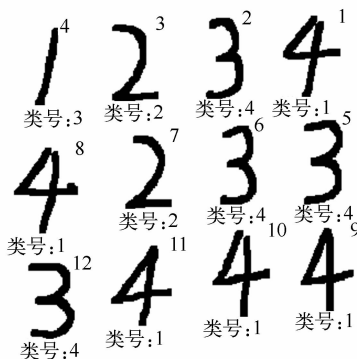
如图 9-16(a)所示,对待聚类样品进行聚类时,选择不同的距离类型,在输入对话框中输入类中心数,K 均值算法最大迭代次数,退火次数和降温速度,如图 9-16(b)所示,最终输出聚类划分结果。如图 9-16(c)所示,最优解出现时的退火次数,如图 9-16(d)所示。如图 9-16(e)所示为整个退火过程中输出的最优目标函数值,可以看出 K 均值聚类的最终结果并不是全局最优,经过逐次退火,最终在第 16 次退火时找到全局最优解。



(a) 待聚类样品



(b) 输入参数



(c) 聚类效果图



(d) 最优解出现时的退火次数

K均值算法, 最优目标函数值: 11.8608
 已退火1次; 最优目标函数值: 11.8608
 已退火2次; 最优目标函数值: 11.8608
 已退火3次; 最优目标函数值: 11.8608
 已退火4次; 最优目标函数值: 11.8608
 已退火5次; 最优目标函数值: 11.8608
 已退火6次; 最优目标函数值: 11.8608
 已退火7次; 最优目标函数值: 11.8608
 已退火8次; 最优目标函数值: 11.5977
 已退火9次; 最优目标函数值: 11.5977
 已退火10次; 最优目标函数值: 11.5977
 已退火11次; 最优目标函数值: 11.5977
 已退火12次; 最优目标函数值: 11.5977
 已退火13次; 最优目标函数值: 11.5977
 已退火14次; 最优目标函数值: 11.5977
 已退火15次; 最优目标函数值: 11.5977
 已退火16次; 最优目标函数值: 10.8963
 已退火17次; 最优目标函数值: 10.8963
 已退火18次; 最优目标函数值: 10.8963
 已退火19次; 最优目标函数值: 10.8963
 已退火20次; 最优目标函数值: 10.8963

(e) 退火过程

图 9-16 基于模拟退火思想的改进 K 均值聚类算法效果图

本章小结

本章介绍了两种基于试探的未知类别聚类算法,包括最临近规则的试探法和最大最小距离算法,还介绍了五种层次聚类算法,包括最短距离法、最长距离法、中间距离法、重心法、类平均距离法;介绍了两种动态聚类算法, K 均值算法和迭代自组织的数据分析算法(ISODATA),最后介绍了模拟退火算法以及基于模拟退火思想的改进 K 均值算法。

习题9

1. 样品间的距离度量方式有哪些?
2. 简述基于试探的未知类别聚类算法。
3. 什么是层次聚类算法?它与基于试探的未知类别聚类算法有何异同?
4. 简述 K 均值算法的基本思想。
5. 叙述 ISODATA 的计算步骤。
6. 简述模拟退火算法的基本原理。

第 10 章 模糊聚类分析

本章要点:

- ☑ 模糊集的基础概念
- ☑ 模糊集运算
- ☑ 模糊关系
- ☑ 模糊集在模式识别中的应用
- ☑ 基于模糊的聚类分析

1965 年美国加利福尼亚州大学自动控制专家 C. A. Zadeh 教授首次发表“Fuzzy Sets”的论文,从此一门研究事物模糊性的新兴学科——模糊数学便应运而生。在信息科学、系统工程、生物科学、社会科学、心理学、医学等方面都有广泛的应用。

模糊集理论是对传统集合理论的一种推广,在传统集合理论中,一个元素或者属于一个集合,或者不属于一个集合;而对于模糊集来说,每一个元素都是以一定的程度属于某个集合,也可以同时以不同的程度属于几个集合。精确性与模糊性相对立,是当今科学发展所面临的一个十分突出的矛盾。各门学科迫切要求数字化、定量化,但科学的深入发展意味着研究对象的复杂化,而复杂的东西又往往难以精确化,电子计算机的出现,可以在一定程度上解决了这个矛盾,然而计算机的应用也更深刻地暴露了这个矛盾,原先的基于数学公式基础上的程序,要求高度的精确,但机器所执行的日益繁杂的任务,往往无法实现高度的精确。当一个系统复杂性增大时,其精确化的能力将降低。当达到一定的阈值限度时,复杂性和精确度将互相排斥。因此,如何使用模糊信息进行处理,使计算机带有接近人类的智能,这对简化模式识别系统使其更加实用可靠,无疑带有战略性的科研课题。

人对于客观事物的认识往往带有模糊性,例如,常说“年轻”、“老年”、“高矮”、“胖瘦”等都是带有模糊性的语言,人类大都用这些模糊的词语来交流思想,互通信息,然后进行推理分析、综合判断,最后作出决策。客观事物是有确定性的,而反映在人的认识上却带有模糊性,人对于客观事物的识别往往只通过一些模糊信息的综合,便可以获得足够精确的定论,例如,从远方走来一个人,如果对这个人比较熟悉,只要从来人的“高矮”、“胖瘦”、“穿戴特点”等模糊信息,就可以判断“大概是 XX”。模式识别这门学科的目的是把人类大脑的感觉、分析推理、决策分类等能力用计算机来加以实现。因此,如何用模糊信息进行处理,使计算机接近人类的智能,是当前非常重要的研究课题。

10.1 模糊集的基本概念

1. 隶属函数

集合可以表现概念,把具有某种属性的元素的全体叫做集合,把集合里的每个成员叫做这个集合的元素。普通集合论中的元素对集合表现的是绝对隶属关系,例如,从一群人(\tilde{X})中,挑选出所有的男人来,构成子集 ω_A ,任意讨论 ω_A 中的一个人 X , X 与 ω_A 之间只有“属于”和

“不属于”两类关系,绝不能模棱两可。

但是如果要把这群人中的中年人找出来,构成子集 ω_A ,如图 10-1(a)所示。就不能对 \tilde{X} 中的一个人 X 用“是”“否”来作出肯定回答,在“是”“否”中间容许有中间状态,提出隶属程度的思想,用隶属函数来代替普通集合论中的特征函数,把这群中年人构成一个模糊子集 ω_A ,如图 10-1(b)所示,用隶属函数 μ 来刻画每个人隶属于“中年人”的程度。

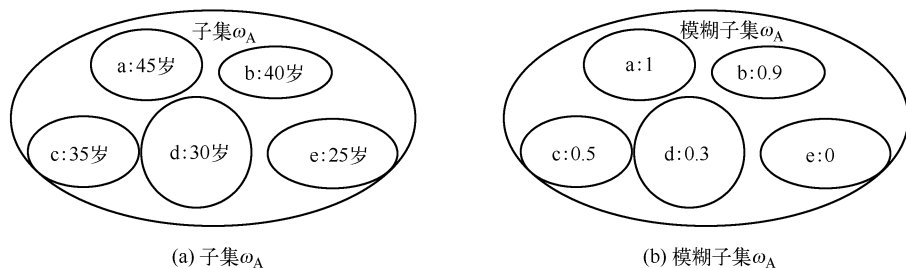


图 10-1 子集 ω_A 与模糊子集 ω_A 之间的关系

隶属函数是表示一个对象 X 隶属于集合 ω_A 的程度的函数,通常记做 $\mu_{\omega_A}(X)$,其自变量范围是有可能属于集合 ω_A 的对象(即集合 ω_A 所在空间中的所有点),取值范围是 $[0, 1]$,即 $0 \leq \mu_{\omega_A}(X) \leq 1$ 。 $\mu_{\omega_A}(X) = 1$ 表示向完全属于 ω_A ,相当于传统集合概念上的 $X \in \omega_A$;而 $\mu_{\omega_A}(X) = 0$ 表示 X 完全不属于集合 ω_A ,相当于传统集合概念上的 $X \notin \omega_A$ 。

对每个元素指定一个对于 ω_A 的隶属程度,45 岁的 a 隶属中年人的程度为 1,40 岁的 b 隶属中年程度为 0.9,35 岁的 c 隶属中年人的程度为 0.5,30 岁的 d 隶属中年人的程度为 0.3,25 岁的 e 肯定不是中年人,隶属中年程度为 0。

隶属度是模糊集合赖以建立的基石,要确定恰当的隶属函数并不容易,迄今仍无一个统一标准的法则可以遵循。这需要对被描述的概念有足够的了解,一定的数学技巧,而且还包括心理测量的进行与结果的运用。正如某一事件的发生与否有一定的不确定性(随机性)一样,某一对象是否符合某一概念也有一定的不确定性(称之为模糊性)。

随机性是因果律的一种或缺,在那里事件本身具有明确的含义,只是由于条件不完全,使得在条件与事件间不能出现决定性的因果关系。概率论的运用,得以从随机性中把握广义的因果律——隶属规律。

模糊性则是排中律的一种或缺,在这里由于概念本身没有明确的外延,故而某一对象是否符合这一概念的划分,就有不确定性,模糊数学正是从这一不确定性(模糊性)中确立的广义的排中律——隶属规律。

隶属度的具体确定,往往包含着人脑的加工,包含着某一种心理过程,但心理过程也是物质性的,心理物理学的大量实验已经表明:人由各种感觉获得心理量与外界刺激的物理量间,保持着相当严格的关系,对心理测量结果的运用与修正,导致隶属度的正确建立。

在某些场合,隶属度可以用模糊统计的方法来确定。模糊统计实验,有四个要素:

① 论域 \tilde{X} ,如手写数字集合;

② \tilde{X} 中的一个元素 X_0 ,如某一个手写的数字“2”;

③ \tilde{X} 中的一个边界可变的普通集合 ω_A ,如“写得像‘2’的手写数字”。 ω_A 联系于一个模糊集及相应的模糊概念。

④ 条件 s , 它联系着按模糊概念所进行的划分过程的全部主客观因素, 它制约着 ω_A 边界的改变。例如, 不同的实验者对“该手写数字像不像数字‘2’”的理解。

模糊性产生的根本原因是: s 对按模糊概念所做的划分引起 ω_A 的变异, 它可能覆盖了 X_0 , 也可不覆盖, 这就导致 X_0 对 ω_A 的隶属关系不确定。例如, 有的人认为该手写数字像“2”, 有的人认为不像。

模糊统计实验的基本要求是在每一次实验下, 要对 X_0 是否属于 ω_A 做出一个确切的判断, 做 N 次实验, 就可以算出 X_0 对 ω_A 的隶属频率:

$$X_0 \text{ 对 } \omega_A \text{ 的隶属频率} = \frac{\text{“} X_0 \in \omega_A \text{” 的次数}}{N}$$

其他隶属度函数确定的方法还有二元对比排序法、推理法、专家评分法等。

2. 模糊子集的定义

定义 设 $\tilde{X} = \{X\}$ 是一个集合, $\mu_{\omega_A}(X)$ 定义在 \tilde{X} 上, 并且 $\mu_{\omega_A}(X) \in [0, 1]$, 则给定论域 \tilde{X} 上的一个模糊子集 ω_A , 是指: 对于任意 $X \in \tilde{X}$, 都确定了一个数 $\mu_{\omega_A}(X)$, 称 $\mu_{\omega_A}(X)$ 为 X 对 ω_A 的隶属度。 $\mu \rightarrow \mu_{\omega_A}(X)$ 叫做 ω_A 的隶属函数。模糊子集完全由其隶属函数所刻画。当 $\mu_{\omega_A}(X)$ 的值域为 $\{0, 1\}$ 时, μ_{ω_A} 蜕化为一个普通子集的特征函数, 普通子集是模糊子集的特殊形态。参见图 10-1, 在论域 \tilde{X} 中确定一个模糊子集 ω_A , 它表示“中年人”这一模糊概念。即 $\tilde{X} = \{a, b, c, d, e\}$, 对于每一个元素, 制定一个对于 ω_A 的隶属程度:

$$a \rightarrow 1, b \rightarrow 0.9, c \rightarrow 0.5, d \rightarrow 0.3, e \rightarrow 0$$

这样便确定了一个模糊子集 ω_A , 它是“中年人”这一模糊概念在论域 \tilde{X} 上的表现。这里, \tilde{X} 由有限个元素组成, 叫做有限论域。有限论域上的模糊子集, 可以用向量来表示, “中年人”在 \tilde{X} 上的模糊子集 ω_A , 可以写成

$$\omega_A = (1, 0.9, 0.5, 0.3, 0)$$

也可以采用 Zadeh 的记法:

$$\omega_A = 1/a + 0.9/b + 0.5/c + 0.3/d + 0/e$$

或

$$\omega_A = 1/a + 0.9/b + 0.5/c + 0.3/d$$

在此不要误把上式右端当做分式求和。分母位置放置的是论域 \tilde{X} 中的元素, 分子位置放置的是相应元素的隶属度。当隶属度为 0 时, 可以不放置此项。也可以采用另一种记法:

$$\omega_A = \{(1, a), (0.9, b), (0.5, c), (0.3, d), (0, e)\}$$

模糊子集是通过隶属函数来定义的。如果要问 ω_A 究竟是由哪些元素组成的? 那么我们必须对隶属度取一定的阈值 α , 当选取 α 不同时, 其模糊集的范围也不同。如上例中有

$$\omega_{A_1} = \{a\}, \omega_{A_{0.9}} = \{a, b\}, \omega_{A_{0.5}} = \{a, b, c\}, \omega_{A_{0.3}} = \{a, b, c, d\}, \omega_{A_0} = \{a, b, c, d, e\}$$

10.2 模糊集运算

10.2.1 模糊子集运算

两个模糊子集间的运算, 实际上就是逐点对隶属度做相应的运算。

(1) 两模糊子集的相等

设 ω_A, ω_B 均为 \tilde{X} 中的模糊集, 若对 $\forall X \in \tilde{X}$ 均有 $\mu_{\omega_A}(X) = \mu_{\omega_B}(X)$, 则称 ω_A 和 ω_B 相等, 即

$$\omega_A = \omega_B \Leftrightarrow \mu_{\omega_A}(X) = \mu_{\omega_B}(X) \quad (10-1)$$

(2) 包含

设 ω_A 与 ω_B 均为 \tilde{X} 中的模糊集, 若对 $\forall X \in \tilde{X}$ 均有 $\mu_{\omega_A}(X) \leq \mu_{\omega_B}(X)$, 则称 ω_B 包含 ω_A , 即

$$\omega_A \subseteq \omega_B \Leftrightarrow \mu_{\omega_A}(X) \leq \mu_{\omega_B}(X) \quad (10-2)$$

(3) 空集

设 ω_A 为 \tilde{X} 中的模糊集, 若对 $\forall X \in \tilde{X}$ 均有 $\mu_{\omega_A}(X) = 0$, 则称 ω_A 为空集, 即

$$\omega_A = \emptyset \Leftrightarrow \mu_{\omega_A}(X) = 0 \quad (10-3)$$

(4) 补集

设 $\omega_A, \overline{\omega_A}$ 为 \tilde{X} 中的模糊集, 若对于 $\forall X \in \tilde{X}$ 均有 $\mu_{\omega_A}(X) = 1 - \mu_{\overline{\omega_A}}(X)$, 则称 $\overline{\omega_A}$ 为 ω_A 的补集, 即

$$\overline{\omega_A} \Leftrightarrow \mu_{\overline{\omega_A}}(X) = 1 - \mu_{\omega_A}(X) \quad (10-4)$$

(5) 全集

设 ω_A 为 \tilde{X} 中的模糊子集, 若对于 $\forall X \in \tilde{X}$ 均有 $\mu_{\omega_A}(X) = 1$, 则称 ω_A 为全集, 记做 Ω , 即

$$\omega_A = \Omega \Leftrightarrow \mu_{\omega_A}(X) = 1 \quad (10-5)$$

(6) 并集

设 $\omega_A, \omega_B, \omega_C$ 均为 \tilde{X} 中的模糊集, 若对于 $\forall X \in \tilde{X}$ 均有 $\mu_{\omega_C}(X) = \max\{\mu_{\omega_A}(X), \mu_{\omega_B}(X)\}$, 则称 ω_C 为 ω_A 与 ω_B 的并集, 即

$$\omega_C = \omega_A \cup \omega_B \Leftrightarrow \mu_{\omega_C}(X) = \max\{\mu_{\omega_A}(X), \mu_{\omega_B}(X)\} \quad (10-6)$$

(7) 交集

设 $\omega_A, \omega_B, \omega_C$ 均为 \tilde{X} 中的模糊集, 若对于 $\forall X \in \tilde{X}$ 均有 $\mu_{\omega_C}(X) = \min\{\mu_{\omega_A}(X), \mu_{\omega_B}(X)\}$, 则称 ω_C 为 ω_A 与 ω_B 的交集, 即

$$\omega_C = \omega_A \cap \omega_B \Leftrightarrow \mu_{\omega_C}(X) = \min\{\mu_{\omega_A}(X), \mu_{\omega_B}(X)\} \quad (10-7)$$

例: 设有两个模糊集 ω_A, ω_B ,

$$\omega_A = 1.0/a + 0.9/b + 0.6/c + 0.2/d + 0.3/e + 0.0/f$$

$$\omega_B = 0.9/a + 0.8/b + 0.7/c + 0.3/d + 0.2/e + 0.1/f$$

则

$$\overline{\omega_A} = 0.0/a + 0.1/b + 0.4/c + 0.8/d + 0.7/e + 1.0/f$$

$$\overline{\omega_B} = 0.1/a + 0.2/b + 0.3/c + 0.7/d + 0.8/e + 0.9/f$$

$$\omega_A \cap \omega_B = 0.9/a + 0.8/b + 0.6/c + 0.2/d + 0.2/e + 0.0/f$$

$$\omega_A \cup \omega_B = 1.0/a + 0.9/b + 0.7/c + 0.3/d + 0.3/e + 0.1/f$$

10.2.2 模糊集运算性质

在普通集合中成立的各种基本性质,一般的对于模糊集也都成立。但由于在模糊集中,一般的互补律不成立,因而需要注意到虽然模糊集在包含关系上构成分配格,但并未构成布尔格。各种基本性质如下:

$$\omega_A \subseteq \omega_A \text{ (自反律)} \quad (10-8)$$

$$\text{若 } \omega_A \subseteq \omega_B, \omega_B \subseteq \omega_A, \text{ 则 } \omega_A = \omega_B \text{ (反对称律)} \quad (10-9)$$

$$\text{若 } \omega_A \subseteq \omega_B, \omega_B \subseteq \omega_C, \text{ 则 } \omega_A \subseteq \omega_C \text{ (传递律)} \quad (10-10)$$

$$\left. \begin{aligned} \omega_A \cup \omega_A &= \omega_A \\ \omega_A \cap \omega_A &= \omega_A \end{aligned} \right\} \text{ (幂等律)} \quad (10-11)$$

$$\left. \begin{aligned} \omega_A \cup \omega_B &= \omega_B \cup \omega_A \\ \omega_A \cap \omega_B &= \omega_B \cap \omega_A \end{aligned} \right\} \text{ (交换律)} \quad (10-12)$$

$$\left. \begin{aligned} (\omega_A \cup \omega_B) \cup \omega_C &= \omega_A \cup (\omega_B \cup \omega_C) \\ (\omega_A \cap \omega_B) \cap \omega_C &= \omega_A \cap (\omega_B \cap \omega_C) \end{aligned} \right\} \text{ (结合律)} \quad (10-13)$$

$$\left. \begin{aligned} (\omega_A \cap \omega_B) \cup \omega_A &= \omega_A \\ (\omega_A \cup \omega_B) \cap \omega_A &= \omega_A \end{aligned} \right\} \text{ (吸收律)} \quad (10-14)$$

$$\left. \begin{aligned} \omega_A \cup (\omega_B \cap \omega_C) &= (\omega_A \cup \omega_B) \cap (\omega_A \cup \omega_C) \\ \omega_A \cap (\omega_B \cup \omega_C) &= (\omega_A \cap \omega_B) \cup (\omega_A \cap \omega_C) \end{aligned} \right\} \text{ (分配律)} \quad (10-15)$$

$$\overline{\overline{\omega_A}} = \omega_A \text{ (双重否定律)} \quad (10-16)$$

$$\left. \begin{aligned} \overline{(\omega_A \cup \omega_B)} &= \overline{\omega_A} \cap \overline{\omega_B} \\ \overline{(\omega_A \cap \omega_B)} &= \overline{\omega_A} \cup \overline{\omega_B} \end{aligned} \right\} \text{ (德·摩根律)} \quad (10-17)$$

$$\left. \begin{aligned} \omega_A \cup \Omega &= \Omega, \omega_A \cap \Omega = \omega_A \\ \omega_A \cup \emptyset &= \omega_A, \omega_A \cap \emptyset = \emptyset \end{aligned} \right\} \text{ (定常律)} \quad (10-18)$$

一般地

$$\omega_A \cup \overline{\omega_A} \neq \Omega, \omega_A \cap \overline{\omega_A} \neq \emptyset \text{ (互补律不成立)} \quad (10-19)$$

10.3 模糊关系

本节中将介绍模糊关系的定义、建立及其运算。

1. 模糊矩阵

在现实生活中,我们常要考虑两个模糊集内各元素之间的关系,已知 U, V 是两论域,设 U 是样品甲的状态集, V 是样品乙的状态集。若同时考虑甲、乙两个因素时,则其可能的状态集是由 U, V 中任意搭配的元素对 (u, v) 构成。称为 U 与 V 的笛卡儿乘积集,记做

$$U \times V = \{(u, v) \mid u \in U, v \in V\} \quad (10-20)$$

笛卡儿乘积集是两集合元素间的无约束搭配。若对这种搭配加以一定的限制,便表现了 U, V 之间的某种特殊关系,称为 U, V 的模糊关系 R 。模糊关系 R 的隶属函数 $\mu_R(U, V)$ 叫做 (U, V)

具有关系 R 的程度, $\mu_R: U \times V \rightarrow [0, 1]$, 表示 U, V 具有关系 R 的程度。

若有一批样品共有 N 个, 每个样品有 n 个特征, 则可把 X 看做一个 n 维列行向量, 该向量 X 称为特征向量, 样品 i 记做

$$X_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{in} \end{pmatrix} = (x_{i1}, x_{i2}, \dots, x_{in})^T$$

如表 10-1 所示为原始资料矩阵

表 10-1 原始资料矩阵

特征 样品	x_1	x_2	\dots	x_k	\dots	x_n
X_1	x_{11}	x_{12}	\dots	x_{1k}	\dots	x_{1n}
X_2	x_{21}	x_{22}	\dots	x_{2k}	\dots	x_{2n}
\dots	\dots	\dots	\dots	\dots	\dots	\dots
X_i	x_{i1}	x_{i2}	\dots	x_{ik}	\dots	x_{in}
\dots	\dots	\dots	\dots	\dots	\dots	\dots
X_j	x_{j1}	x_{j2}	\dots	x_{jk}	\dots	x_{jn}
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
X_N	x_{N1}	x_{N2}	\dots	x_{Nk}	\dots	x_{Nn}

样品 i 与样品 j 的之间的相似程度, $i = 1, 2, \dots, N, j = 1, 2, \dots, N, i$ 与 j 任意搭配, 表示它们之间的贴近程度, 构成模糊关系 R , 此时模糊关系 R 可以用矩阵形式表示, 即

$$R = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1N} \\ r_{21} & r_{22} & \dots & r_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ r_{N1} & r_{N2} & \dots & r_{NN} \end{bmatrix}$$

$R = (r_{ij})$, 其中 $r_{ij} = \mu_R(X_i, X_j)$

显然有 $0 \leq r_{ij} \leq 1, (1 \leq i, j \leq N)$, 满足以上条件的矩阵称为模糊矩阵。特别地, 当 $r_{ij} \in \{0, 1\}, (1 \leq i, j \leq N)$ 时, 矩阵 R 退化为布尔矩阵。布尔矩阵可以表达一种普通的关系。

2. 模糊关系的建立

模糊关系可以用于聚类分析, 但对样品的聚类效果怎样? 关键是选择合理的统计指标, 即被选中的指标应有明确的实际意义, 有较强的分辨力和代表性。在统计指标确定后就可以按照下面的内容进行分类。

① 把各代表点的统计指标的数据标准化, 以便分析和比较。这一步也称正规化。为把标准化数据压缩到 $(0, 1)$ 闭区间, 可以用极值标准化公式:

$$x = \frac{x' - x'_{\min}}{x'_{\max} - x'_{\min}} \quad (10-21)$$

当 $x' = x'_{\max}$ 时, $x = 1$; 当 $x' = x'_{\min}$ 时, $x = 0$, 否则取 $[0, 1]$ 之间。

② 算出被分类对象间具有此种关系的程度 r_{ij} (最通常是 X_i 与 X_j 的相似程度), 其中 $i, j =$

1, 2, ..., N。N 为对象个数。从而确定论域 \tilde{X} 上的模糊关系 R。

3. 计算 r_{ij} 常用的方法

(1) 欧氏距离法

$$r_{ij} = 1 - \sqrt{\frac{1}{n} \sum_{k=1}^n (x_{ik} - x_{jk})^2} \quad (10-22)$$

式中, x_{ik} 为第 i 个样品第 k 个特征的值, x_{jk} 为第 j 个样品的第 k 个特征的值。

(2) 数量积

$$r_{ij} = \begin{cases} 1, & \text{当 } i = j \\ \sum_{k=1}^n \frac{x_{ik}x_{jk}}{M}, & \text{当 } i \neq j \end{cases} \quad (10-23)$$

式中, M 为一适当选择的正数, 满足 $M \geq \max_{ij} \left(\sum_{k=1}^n x_{ik}x_{jk} \right)$

(3) 相关系数

$$r_{ij} = \frac{\sum_{k=1}^n |x_{ik} - \bar{x}_k| |x_{jk} - \bar{x}_k|}{\sqrt{\sum_{k=1}^n (x_{ik} - \bar{x}_k)^2} \sqrt{\sum_{k=1}^n (x_{jk} - \bar{x}_k)^2}} \quad \text{其中 } \bar{x}_k = \frac{1}{N} \sum_{p=1}^N x_{pk} \quad (10-24)$$

(4) 指数相似系数

$$r_{ij} = \frac{1}{n} \sum_{k=1}^n \exp\left(\frac{3}{4} \frac{(x_{ik} - x_{jk})^2}{\delta_k^2}\right) \quad \text{其中 } \delta_k^2 \text{ 为适当选择的正数} \quad (10-25)$$

(5) 非参数法

令 $x'_{ik} = x_{ik} - \bar{x}_k$

$n^+ = \{x'_{im} \cdot x'_{jm}\}$ 中大于 0 的个数

$n^- = \{x'_{im} \cdot x'_{jm}\}$ 中小于 0 的个数

$$r_{ij} = \frac{|n^+ - n^-|}{n^+ + n^-} \quad (10-26)$$

(6) 最大最小法

$$r_{ij} = \frac{\sum_{k=1}^n \min(x_{ik}, x_{jk})}{\sum_{k=1}^n \max(x_{ik}, x_{jk})} \quad (10-27)$$

(7) 算数平均法

$$r_{ij} = \frac{\sum_{k=1}^n \min(x_{ik}, x_{jk})}{\frac{1}{2} \sum_{k=1}^n (x_{ik} + x_{jk})} \quad (10-28)$$

(8) 几何平均最小法

$$r_{ij} = \frac{\sum_{k=1}^n \min(x_{ik}, x_{jk})}{\sum_{k=1}^n \sqrt{(x_{ik} \cdot x_{jk})}} \quad (10-29)$$

(9) 绝对值指数法

$$r_{ij} = \exp\left(-\sum_{k=1}^n |x_{ik} - x_{jk}|\right) \quad (10-30)$$

(10) 绝对值倒数法

$$r_{ij} = \begin{cases} 1, & \text{当 } i = j \\ \frac{M}{\sum_{k=1}^n |x_{ik} - x_{jk}|}, & \text{当 } i \neq j \end{cases} \quad (10-31)$$

(11) 绝对值减数法

$$r_{ij} = \begin{cases} 1, & \text{当 } i = j \\ 1 - M \sum_{k=1}^n |x_{ik} - x_{jk}|, & \text{当 } i \neq j \end{cases} \quad \text{其中 } M \text{ 适当选取, 使 } 0 \leq r_{ij} \leq 1 \quad (10-32)$$

(12) 主观评定法

以百分制打分, 然后除以 100, 得到 $[0, 1]$ 区间的一个数, 也可多人打分求平均。

上述各种方法的优劣不能一概而论, 模糊关系用于分类时, 设计并选取建立模糊关系的方法是重要的。

4. 模糊矩阵的运算

(1) 并、交、非运算的定义

定义 设以 F_{nm} 表示全体 n 行 m 列的模糊矩阵, 对任意 $R, S \in F_{nm}$, 有 $R = (r_{ij}), S = (s_{ij})$, 定义

$$R \cup S = (r_{ij} \vee s_{ij})$$

$$R \cap S = (r_{ij} \wedge s_{ij})$$

$$\bar{R} = (1 - r_{ij})$$

分别为 R 与 S 的并、交、及 R 的余矩阵。

例: 设

$$R = \begin{pmatrix} 0.5 & 0.3 \\ 0.4 & 0.8 \end{pmatrix} \quad S = \begin{pmatrix} 0.8 & 0.5 \\ 0.3 & 0.7 \end{pmatrix}$$

$$\text{则 } R \cup S = \begin{pmatrix} 0.8 & 0.5 \\ 0.4 & 0.8 \end{pmatrix}, R \cap S = \begin{pmatrix} 0.5 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}, \bar{R} = \begin{pmatrix} 0.5 & 0.7 \\ 0.6 & 0.2 \end{pmatrix}$$

如果 $r_{ij} = s_{ij}$ 对所有 i, j 成立, 则称 $R = S$ 。

单位矩阵 I : 对角线上为 1, 其余为 0 的模糊矩阵; 零矩阵: 全部项为 0 的模糊矩阵, 以 0 表

示;全矩阵 E :全部项均为 1 的模糊矩阵。

对任意 $R, S, T \in F_{nm}$, 下列性质成立:

性质 1 交换律。

性质 2 结合律。

性质 3 分配律。

性质 4 等幂律。

性质 5 吸收律。

性质 6 $0 \cup R = R, 0 \cap R = 0; E \cup R = E, E \cap R = R$ 。

性质 7 $0 \subseteq R \subseteq E$ 。

如果对任意 i, j 均有 $r_{ij} \leq s_{ij}$, 称矩阵 S 包含矩阵 R , 记做 $R \subseteq S$ 。

性质 8 $R \subseteq S \Leftrightarrow R \cup S = S \Leftrightarrow R \cap S = R$ 。

性质 9 若 $R_1 \subseteq S_1, R_2 \subseteq S_2$, 则有 $(R_1 \cup R_2) \subseteq (S_1 \cup S_2), (R_1 \cap R_2) \subseteq (S_1 \cap S_2)$ 。

性质 10 $R \subseteq S \Leftrightarrow R \supseteq \bar{S}$ 。

对于任意 $\lambda \in [0, 1]$, 记 $R_\lambda = (\lambda_{ij})$, 其中

$$\lambda_{ij} = \begin{cases} 1, & \text{当 } r_{ij} \geq \lambda \\ 0, & r_{ij} < \lambda \end{cases}$$

称 R_λ 为 R 的 λ 截矩阵, 它所对应的关系叫做 R 的截关系。截矩阵必是布尔矩阵。

性质 11 $R \subseteq S \Leftrightarrow R_\lambda \subseteq S_\lambda$ 。

性质 12 $(R \cup S)_\lambda = R_\lambda \cup S_\lambda$ (此时不能推广到无限个矩阵的并集)

$$(R \cap S)_\lambda = R_\lambda \cap S_\lambda$$

(2) 模糊关系的合成与模糊矩阵的合成

设 $Q \in F_{U \times V}, R \in F_{V \times W}$, 称 Q 对 R 的合成为 U 到 W 的一个模糊关系, 它具有隶属度函数

$$\mu_{(Q \circ R)}(x, z) = \bigvee_{y \in V} (\mu_Q(x, y) \wedge \mu_R(y, z))$$

设 $Q = (q_{ij})_{nm}, R = (r_{jk})_{m1}$, 定义 $S = Q \circ R \in F_{n1}$ 且有

$$S_{ik} = \bigvee_{j=1}^m (q_{ij} \wedge r_{jk})$$

式中, “ \wedge ”表示求最小值, “ \vee ”表示求最大值。

称矩阵 Q 对矩阵 R 的合成为矩阵 S 。 S 也叫做 Q 对 R 的模糊积。

有上述定义可以得到以下结论:对于有限论域, 由模糊矩阵乘积的定义, 其运算过程与普通矩阵乘法相同, 只不过将实数加法改成求并集, 实数乘法改成求交集而已。

$$\text{例: 若 } Q = \begin{bmatrix} 0.3 & 0.7 & 0.2 \\ 1 & 0 & 0.4 \\ 0 & 0.5 & 1 \\ 0.6 & 0.7 & 0.8 \end{bmatrix} \quad R = \begin{bmatrix} 0.1 & 0.9 \\ 0.9 & 0.1 \\ 0.6 & 0.4 \end{bmatrix} \quad \text{则有 } S = Q \circ R = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.9 \\ 0.6 & 0.4 \\ 0.7 & 0.6 \end{bmatrix}$$

$$\begin{aligned} S_{11} &= (q_{11} \wedge r_{11}) \vee (q_{12} \wedge r_{21}) \vee (q_{13} \wedge r_{31}) \\ &= (0.3 \wedge 0.1) \vee (0.7 \wedge 0.9) \vee (0.2 \wedge 0.6) \\ &= 0.1 \vee 0.7 \vee 0.2 \\ &= 0.7 \end{aligned}$$

如此,等等。

性质 13 对模糊矩阵有

$$(Q \circ R)_\lambda = Q_\lambda \circ R_\lambda$$

性质 14 模糊乘法结合律

$$(Q \circ R) \circ S = Q \circ (R \circ S)$$

推论: $R^m \circ R^n = R^{m+n}$

性质 15

$$(Q \cup R) \circ S = (Q \circ S) \cup (R \circ S)$$

$$S \circ (Q \cup R) = (S \circ Q) \cup (S \circ R)$$

性质 16 $0 \circ R = R \circ 0, I \circ R = R \circ I = R$

(3) 模糊关系的基本性质

我们把具有自反性、对称性、传递性的模糊关系称为等价模糊关系。

自反性: 对于一个 $\tilde{X} \times \tilde{X}$ 中的模糊关系 R , 对所有 $X \in \tilde{X}$, 若 $\mu_R(X, X) = 1$ 成立, 则称这种模糊关系 R 具有自反性。

对称性: 若对所有的 $(X_i, X_j) \in \tilde{X} \times \tilde{X}$ 均有 $\mu_R(X_i, X_j) = \mu_R(X_j, X_i)$ 成立, 则称模糊关系 R 具有对称性。

相似关系: 我们把具有自反性和对称性的模糊关系称为相似关系。

一个 $\tilde{X} \times \tilde{X}$ 中的模糊关系 R , 如果矩阵 $\mu_R(X_i, X_j)$ 中的对角线元素均为 1, 则 R 具有自反性, 若主对角线对称的元素均相等, 则 R 具有对称性。若满足 $R \cdot R \subseteq R$, 则 R 具有传递性, 传递性关系实质它包含着它与它自己的合成。

例如:

$$R = \begin{bmatrix} 1 & 0.9 & 0.6 & 0.3 & 0.3 \\ 0.9 & 1 & 0.9 & 0.9 & 0.9 \\ 0.6 & 0.9 & 1 & 0.3 & 0.3 \\ 0.3 & 0.9 & 0.3 & 1 & 0.2 \\ 0.3 & 0.9 & 0.3 & 0.2 & 1 \end{bmatrix}$$

R 对角线为 1, 具有自反性; R 关于对角线对称, 具有对称性; $R \cdot R = R$, 满足传递性, 所以, R 是一个等价矩阵。

10.4 模糊集在模式识别中的应用

模式识别实际上是一个分类问题, 它是将一个未知模式指定为已知类别中的一种。利用模糊子集理论可以归纳为: 隶属原则识别法和择近原则识别法。在实际问题中还有另一种分类问题, 也就是我们可以利用模糊关系进行聚类分析。

1. 隶属原则识别法

在通常的模式识别中, 所谓模式, 总是有一个明确、清晰、肯定的样式, 例如, 识别手写数字时, 其模式就是印刷体的数字。根据隶属度最大的原则来分类是很自然的。这种直接由计算样品的隶属度来判断其归属的方法, 称做模式分类的隶属度原则, 也叫做模糊模式分类的直接

方法。

隶属度原则：设论域 \tilde{X} 中有 M 个模糊子集 $\omega_1, \omega_2, \dots, \omega_M$, 且对于每一个 ω_i 都有隶属度函数 $\mu_{\omega_i}(X)$, 对任意 $X_0 \in \tilde{X}$, 若有

$$\mu_{\omega_i}(X_0) = \max[\mu_{\omega_1}(X_0), \mu_{\omega_2}(X_0), \dots, \mu_{\omega_M}(X_0)] \quad (10-33)$$

则认为 X_0 属于 ω_i , 隶属原则是明显的, 易于公认的, 但其效果如何, 却十分依赖于建立已知模式类隶属函数的技巧。

2. 择近原则识别法

在实际的模式识别中, 被识别的对象往往不是论域 \tilde{X} 中的一个确定的元素, 而是 \tilde{X} 中的一个子集。这时, 所讨论的对象不是一个元素对集合的隶属程度, 而是两个模糊子集间的贴近程度。

设 ω_A, ω_B 是 \tilde{X} 上的两个模糊子集, 则其间的贴近度定义为

$$(\omega_A, \omega_B) = \frac{1}{2}[\omega_A \cdot \omega_B + (1 - \omega_A \odot \omega_B)] \quad (10-34)$$

式中

$$\begin{aligned} \omega_A \cdot \omega_B &= \bigvee_{x \in E} (\omega_A(x) \wedge \omega_B(x)) \\ \omega_A \odot \omega_B &= \bigwedge_{x \in E} (\omega_A(x) \vee \omega_B(x)) \end{aligned}$$

分别称为 A 和 B 的内积和外积。

例如, $\tilde{X} = \{a, b, c, d, e, f\}$

$$\omega_A = 0.6/a + 0.8/b + 1.0/c + 0.8/d + 0.6/e + 0.4/f$$

$$\omega_B = 0.4/a + 0.6/b + 0.8/c + 1.0/d + 0.8/e + 0.6/f$$

则

$$\omega_A \odot \omega_B = (0.6 \vee 0.8) \wedge (0.8 \vee 0.6) \wedge (1.0 \vee 0.8) \wedge (0.8 \vee 1.0) \wedge (0.6 \vee 0.8) \wedge (0.4 \vee 0.6) = 0.6$$

$$\omega_A \cdot \omega_B = (0.6 \wedge 0.4) \vee (0.8 \wedge 0.6) \vee (1.0 \wedge 0.8) \vee (0.8 \wedge 1.0) \vee (0.6 \wedge 0.8) \vee (0.4 \wedge 0.6) = 0.8$$

$$(\omega_A, \omega_B) = \frac{1}{2}[0.8 + (1 - 0.6)] = 0.6$$

利用贴近度进行分类:

设 \tilde{X} 上 n 个模糊子集 $\omega_{A_1}, \omega_{A_2}, \dots, \omega_{A_n}$ 及另一个模糊子集 ω_B , 若有 $1 \leq i \leq n$ 使

$$(\omega_B, \omega_{A_i}) = \max_{1 \leq j \leq n} (\omega_B, \omega_{A_j})$$

则称 ω_B 与 ω_{A_i} 最接近, 或者说 ω_B 属于 ω_{A_j} 类。

上式说明, 要判别某一模糊子集 ω_B 属于 $\omega_{A_1}, \omega_{A_2}, \dots, \omega_{A_n}$ 中的哪一类, 则应首先计算 ω_B 与 $\omega_{A_1}, \omega_{A_2}, \dots, \omega_{A_n}$ 各类的贴近度, 然后把 ω_B 归入贴近度最大的那一类。

10.5 基于模糊的聚类分析

1. 理论基础

同时满足自反性、对称性、传递性的模糊关系是等价模糊关系, 等价模糊关系可以用来进行模式聚类。不具有等价关系的模糊关系不能用于聚类。为了得到模糊关系 R 的模糊等价

关系,可以用 R 自乘得到,即 $R \circ R = R^2, R^2 \circ R^2 = R^4 \cdots$,一直到 $R^{2k} = R^k$,至此, R^k 便是一模糊等价关系。此方法是由“传递闭包”而来的,此处不作证明。

例:设论域 $\tilde{X} = \{X_1, X_2, \cdots, X_5\}$

给定模糊关系矩阵:

$$R = \begin{bmatrix} 1 & 0.48 & 0.62 & 0.41 & 0.47 \\ 0.48 & 1 & 0.48 & 0.41 & 0.47 \\ 0.62 & 0.48 & 1 & 0.41 & 0.47 \\ 0.41 & 0.41 & 0.41 & 1 & 0.41 \\ 0.47 & 0.47 & 0.47 & 0.41 & 1 \end{bmatrix}$$

显然 R 是自反的和对称的。验证可得 $R \circ R = R^2 = R$,故 R 是一个模糊等价关系。现根据不同的 λ 分类:

① $0.62 < \lambda \leq 1$ 时,

$$R_\lambda = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{此时共分五类,每个元素一类,这是“最细分类”。}$$

② $0.48 < \lambda \leq 0.62$ 时,

$$R_\lambda = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{此时分四类: } \{x_1, x_3\}, \{x_2\}, \{x_4\}, \{x_5\}$$

③ 当 $0.47 < \lambda \leq 0.48$ 时,

$$R_\lambda = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{此时分三类: } \{x_1, x_2, x_3\}, \{x_4\}, \{x_5\}$$

④ 当 $0.41 < \lambda \leq 0.47$ 时,

$$R_\lambda = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}, \text{此时分两类: } \{x_1, x_2, x_3, x_5\}, \{x_4\}$$

⑤ $0 < \lambda \leq 0.41$ 时,全部元素分为一类。

2. 实现步骤

① 在左视图建立数据源(手写、标准数字按钮,打开位图文件),如图10-2所示,调用

GetFeature() 函数,获得所有样品特征。

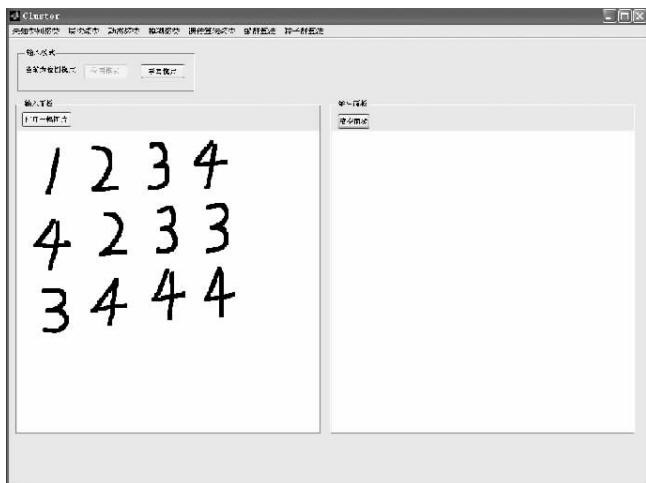


图 10-2 源图像



图 10-3 选择距离模式

② 选择“模糊聚类”菜单,弹出“模糊距离选择”对话框,如图 10-3 所示,从对话框中用户选择距离模式(欧氏距离、数量积、相关系数、最大最小法、算数平均法、几何平均最小法)。

③ 选择不同的聚类计算公式,程序会给出选定距离模式下的模糊矩阵。计算选定距离模式下的模糊矩阵,如图 10-4 所示。

调用 GetFuzzyDistance() 函数,得到样品之间的模糊距离,生成初始模糊矩阵。调用 GetFuzzyIntegral() 函数,得到模糊矩阵中的模糊积。

④ 从对话框输入阈值,如图 10-5 所示。

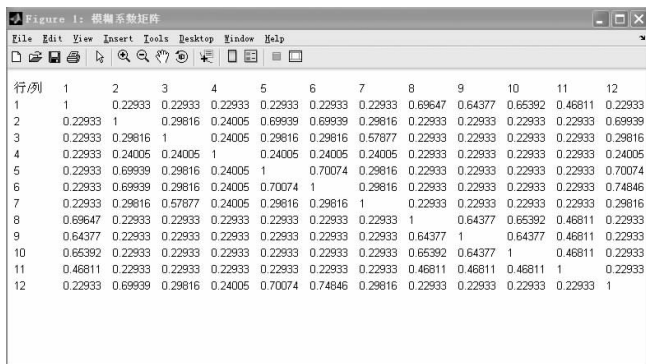


图 10-4 输出在选定距离模式下的模糊矩阵

⑤ 在原来的模糊矩阵中,模糊系数大于给定阈值的将被置成 1,小于阈值的将被置成 0,就会得到根据阈值聚类的模糊矩阵对照表,如图 10-6 所示。

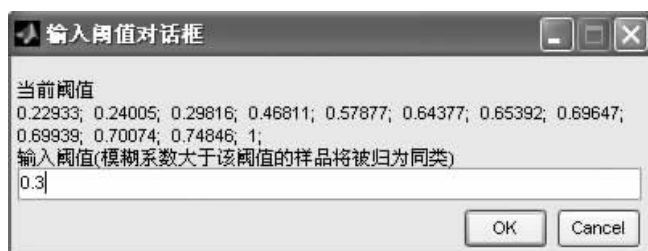


图 10-5 输入聚类模糊系数阈值

⑥ 将等价矩阵中每行为 1 的系数所对应的列归为同一类,输出聚类结果,如图 10-7 所示。

行列	2	3	4	5	6	7	8	9	10	11	12
1	1	0	0	0	0	0	0	1	1	1	0
2	0	1	0	0	1	1	0	0	0	0	1
3	0	0	1	0	0	0	1	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0
5	0	1	0	0	1	1	0	0	0	0	1
6	0	1	0	0	1	1	0	0	0	0	1
7	0	0	1	0	0	0	1	0	0	0	0
8	1	0	0	0	0	0	0	1	1	1	0
9	1	0	0	0	0	0	0	1	1	1	0
10	1	0	0	0	0	0	0	1	1	1	0
11	1	0	0	0	0	0	0	1	1	1	0
12	0	1	0	0	1	1	0	0	0	0	1

图 10-6 输出聚类前后模糊矩阵对照表

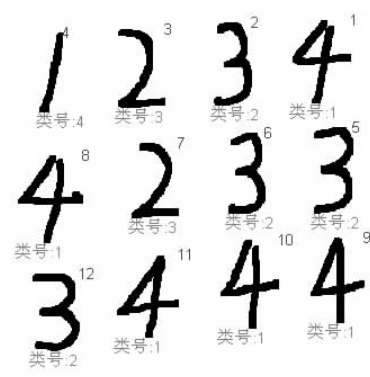


图 10-7 模糊聚类效果

3. 编程代码

函数调用关系:

```
[ m_pattern ] = C_FuzzyCluster( m_pattern, patternNum ) // 用模糊聚类方法对全体样品进行分类
├── [ result ] = GetFuzzyDistance( p1, p2, disType, m_pattern, patternNum ) // 计算两个样品 p1,
│   │   p2 的模糊距离, 距离模式由参数 disType 给出
└── [ result ] = GetFuzzyIntegral( dis, i, j, patternNum ) // 返回模糊距离矩阵 dis 中第 i 行第 j
    │   列的模糊积
    │   %%%%%%%%%%%
    │   % 函数名称: C_FuzzyCluster( )
    │   % 参数: m_pattern: 样品特征库; patternNum: 样品数目
    │   % 返回值: m_pattern: 样品特征库
    │   % 函数功能: 按照模糊聚类法对全体样品进行分类
    │   %%%%%%%%%%%
function [ m_pattern ] = C_FuzzyCluster( m_pattern, patternNum )
    disType = FuzzyDisSelDlg( ) ; % 获得距离计算类型
    dis = zeros( patternNum, patternNum ) ; % 模糊系数矩阵
    tempDis = zeros( patternNum, patternNum ) ;
```

```

% 得到初始模糊距离
for i = 1 : patternNum
    for j = 1 : patternNum
        dis(i,j) = GetFuzzyDistance(i,j,disType,m_pattern,patternNum);
    end
end
% 构造等价类
flag = true;

while(flag)
    flag = false;
    for i = 1 : patternNum
        for j = 1 : patternNum
            if(i == j) % 对角线为 1
                tempDis(i,j) = 1;
            else
                tempDis(i,j) = GetFuzzyIntegral(dis,i,j,patternNum);
            end
        end
    end
    end

    for i = 1 : patternNum
        for j = 1 : patternNum
            if((tempDis(i,j) - dis(i,j))^2 > 0.000001)
                flag = true;
                break;
            end
        end
    end
    if(flag)
        break;
    end
end

dis = tempDis;
end

% 显示模糊系数矩阵
num = zeros(patternNum + 1, patternNum + 1);
for i = 1 : patternNum + 1
    for j = 1 : patternNum + 1
        if(i == 1 && j == 1)
            num(i,j) = j - 1;
        elseif(i == 1 && j == 1)

```

```

        num(i,j) = 0;
    elseif(j == 1)
        num(i,j) = i - 1;
    else
        num(i,j) = dis(i - 1, j - 1);
    end
end
end
% 当样品个数小于 16 个时,显示模糊系数矩阵
if(patternNum < 16)
    a = [(980 - 60 * patternNum)/2, 650 - 30 * patternNum, 65 * patternNum, 30 * patternNum];
    h = figure('Position', a, 'Name', '模糊系数矩阵 ');
    figure(h);
    axes('XTick', [], 'YTick', [], 'Units', 'pixels', 'XColor', [1 1 1], 'YColor', [1 1 1], 'Position',
        [0, 0, a(3), a(4)]);
    for i = 1:patternNum + 1
        for j = 1:patternNum + 1
            if (i ~= 1 || j ~= 1)
                text(60 * (j - 1) + 10, a(4) - 20 * i, num2str(num(i, j)), 'Units', 'pixels');
            else
                text(60 * (j - 1) + 10, a(4) - 20 * i, '行/列 ', 'Units', 'pixels');
            end
        end
    end
end
pointer = 1;
xiShu = -ones(1, patternNum * patternNum);
% 记录模糊系数矩阵中不同的系数
for i = 1:patternNum
    for j = i:patternNum
        done = false;
        for k = 1:pointer
            d = (xiShu(1, k) - dis(i, j))^2;
            if((xiShu(1, k) - dis(i, j))^2 < 0.0000001)
                done = true;
                break;
            end
        end
        xiShu(1, pointer) = dis(i, j);
        pointer = pointer + 1;
    end
end
end

```

```

end

for i = 1:pointer - 1 % 对阈值由小到大排序
    for j = 1:pointer - i - 1
        if(xiShu(1,j) > xiShu(1,j + 1))
            temp = xiShu(1,j);
            xiShu(1,j) = xiShu(1,j + 1);
            xiShu(1,j + 1) = temp;
        end
    end
end

str1 = ['当前阈值 '];
str3 = ['输入阈值(模糊系数大于该阈值的样品将被归为同类)'];
str2 = '';
for i = 1:pointer - 1
    str2 = [str2 num2str(xiShu(1,i)) ' '];
end
show = char(str1, str2, str3);
T = inputdlg(show, '输入阈值对话框 ');
T = str2num(T{1,1});

% 根据阈值输出聚类结果
result = zeros(patternNum, patternNum);
for i = 1:patternNum
    m_pattern(i).category = 0;
    for j = 1:patternNum
        if(dis(i,j) > T)
            result(i,j) = 1;
        else
            result(i,j) = 0;
        end
    end
end

centerNum = 0;

% 显示模糊聚类结果
for i = 1:patternNum + 1
    for j = 1:patternNum + 1
        if(i == 1 && j == 1)
            num(i,j) = j - 1;
        elseif(i == 1 && j == 1)
            num(i,j) = 0;
        elseif(j == 1)

```

```

        num(i,j) = i - 1;
    else
        num(i,j) = result(i - 1,j - 1);
    end
end
end
% 当样品个数小于 16 个时,显示模糊系数矩阵对照表
if(patternNum < 16)
    set(h,'Name','每行中“1”对应的列为同一类','Position',[a(1),a(2),a(3)/2,
    a(4)]);
    axes('XTick',[ ],'YTick',[ ],'Units','pixels','XColor',[1 1 1],'YColor',[1 1 1],'Position',
    [0,0,a(3)/2,a(4)]);
    for i = 1:patternNum + 1
        for j = 1:patternNum + 1
            if (i ~ = 1 || j ~ = 1)
                text(30 * (j - 1) + 10, a(4) - 20 * i, num2str(num(i,j)),
                'Units','pixels');
            else
                text(30 * (j - 1) + 10, a(4) - 20 * i, '行/列','Units','pixels');
            end
        end
    end
    end
    set(h,'WindowStyle','modal');
end

% 按照阈值分类
for i = 1:patternNum
    for j = i:patternNum
        if(result(i,j) == 1)
            if(m_pattern(i).category ~ = 0)
                m_pattern(j).category = m_pattern(i).category;
            elseif(m_pattern(j).category ~ = 0)
                m_pattern(i).category = m_pattern(j).category;
            else
                centerNum = centerNum + 1;
                m_pattern(i).category = centerNum;
                m_pattern(j).category = centerNum;
            end
        end
    end
end
end

% *****
% 函数名称: GetFuzzyDistance( p1,p2,disType,m_pattern,patternNum )

```



```

%参数:          p1: 样品1 序号
%              p2: 样品2 序号
%              disType:          距离模式  1:欧氏距离;2:数量积;
%                                  3:相关系数;4:最大最小法
%                                  5:算数平均法;6:几何平均最小法
%              m _ pattern:样品特征库
%              patternNum:样品数量
%返回值:          result:距离
%  函数功能:计算样品1 和样品2 间的模糊距离,距离模式由参数 disType 给定
%*****
function [ result ] = GetFuzzyDistance( p1 , p2 , disType , m _ pattern , patternNum )
    result = 0;
    global Nwidth;
    switch( disType )
        case 1% 欧氏距离
            max = 0;
            for i = 1 : patternNum - 1
                for j = i + 1 : patternNum
                    tempDis = GetDistance( m _ pattern( i ) , m _ pattern( j ) , 1 ) ;
                    if( max < tempDis )
                        max = tempDis;
                        result = ( max - GetDistance( m _ pattern( p1 ) , m _ pattern( p2 ) , 1 ) ) / max;
                    end
                end
            end
        case 2% 数量积
            max = 0;
            for i = 1 : patternNum - 1
                for j = i + 1 : patternNum
                    temp = m _ pattern( i ) . feature( : ) ' * m _ pattern( j ) . feature( : ) ;
                    if( max < temp )
                        max = temp;
                    end
                end
            end
            temp = m _ pattern( p1 ) . feature( : ) ' * m _ pattern( p2 ) . feature( : ) ;
            result = temp / max;
        case 3% 相关系数
            ap1 = 0;
            ap2 = 0;
            ap1 = mean( mean( m _ pattern( p1 ) . feature ) ) ;
            ap2 = mean( mean( m _ pattern( p2 ) . feature ) ) ;
            ap1 = ap1 * ones( Nwidth * Nwidth , 1 ) ;
    end
end

```

```

ap2 = ap2 * ones( Nwidth * Nwidth,1 );
a = ( m _ pattern( p1 ). feature( : ) - ap1 )' * ( m _ pattern( p2 ). feature( : ) - ap2 );
b1 = ( m _ pattern( p1 ). feature( : ) - ap1 )' * ( m _ pattern( p21 ). feature( : ) - ap1 );
b2 = ( m _ pattern( p2 ). feature( : ) - ap2 )' * ( m _ pattern( p2 ). feature( : ) - ap2 );
if( b2 * b1 ~ = 0 )
    result = a / sqrt( b1 * b2 );
end
case 4% 最大最小法
min = 0;
max = 0;
for m = 1 : Nwidth
    for n = 1 : Nwidth
        if( m _ pattern( p1 ). feature( m , n ) < m _ pattern( p2 ). feature( m , n ) )
            min = min + m _ pattern( p1 ). feature( m , n );
            max = max + m _ pattern( p2 ). feature( m , n );
        else
            min = min + m _ pattern( p2 ). feature( m , n );
            max = max + m _ pattern( p1 ). feature( m , n );
        end
    end
end
if( max ~ = 0 )
    result = min / max;
end
case 5% 算数平均法
min = 0;
max = 0;
for m = 1 : Nwidth
    for n = 1 : Nwidth
        if( m _ pattern( p1 ). feature( m , n ) < m _ pattern( p2 ). feature( m , n ) )
            min = min + m _ pattern( p1 ). feature( m , n );
        else
            min = min + m _ pattern( p2 ). feature( m , n );
        end
        max = max + m _ pattern( p1 ). feature( m , n ) + m _ pattern( p2 ). feature( m , n );
    end
end
if( max ~ = 0 )
    result = 2 * min / max;
end
case 6% 几何平均最小法
min = 0;
max = 0;

```

```

        for m = 1:Nwidth
            for n = 1:Nwidth
                if(m_pattern(p1).feature(m,n) < m_pattern(p2).feature(m,n))
                    min = min + m_pattern(p1).feature(m,n);
                else
                    min = min + m_pattern(p2).feature(m,n);
                end
                max = max + sqrt(m_pattern(p1).feature(m,n) * m_pattern(p2).feature(m,n));
            end
        end
        if(max ~ = 0)
            result = min/max;
        end
    end

%*****
% 函数名称:      GetFuzzyIntegral()
% 参数:          dis: 模糊距离数组
%                i:   数组的第 i 行
%                j:   数组的第 j 列
%                patternNum: 样本数量
% 返回值:        result: 模糊积
% 函数功能:      返回模糊距离数组中第 i 行第 j 列的模糊积
%*****
function [ result ] = GetFuzzyIntegral( dis,i,j,patternNum )
    result = 0;
    for t = 1:patternNum
        if(dis(i,t) < dis(t,j))
            td = dis(i,t);
        else
            td = dis(t,j);
        end
        if(result < td)
            result = td;
        end
    end
end

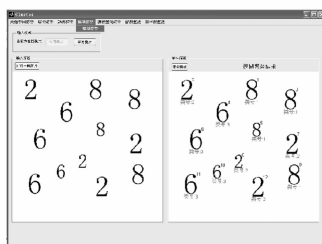
```

4. 效果图

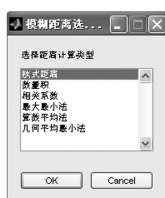
(1) 在选择阈值恰当情况下的聚类效果

在左视图建立数据源(手写、标准数字按钮、打开文件),如图 10-8(a)所示,选择“模糊聚类”

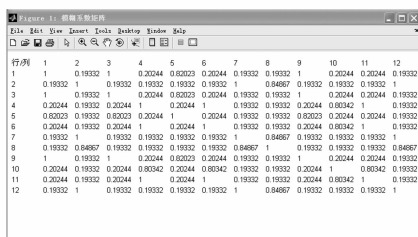
菜单,弹出“模糊距离选择”对话框,如图 10-8(b)所示,读者选择不同的聚类计算公式,然后程序会给出选定距离模式下的模糊矩阵,如图 10-8(c)所示。根据模糊矩阵选取分类阈值,如图 10-8(d)所示,就会得到根据阈值聚类的模糊矩阵对照表。在原来的模糊矩阵中,模糊系数大



(a) 源图像



(b) 选择距离模式



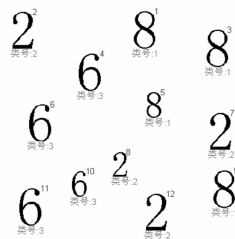
(c) 输出在选定距离模式下的模糊矩阵



(d) 输入聚类模糊系数阈值



(e) 输出聚类前后模糊矩阵对照表



(f) 模糊聚类效果

图 10-8 正确聚类的阈值和结果

于给定阈值的将被置成 1, 小于阈值的将被置成 0, 如图 10-8(e) 所示。可以看到选择不同的分类阈值, 会得到不同的聚类结果, 如图 10-8(f) 所示。不同的聚类计算公式会产生不同的阈值, 而阈值的正确选择, 都能得到正确的聚类结果。将等价矩阵中每行为 1 的系数所对应的列归为同一类, 输出聚类结果。

图 10-8 列举了应用欧氏距离(读者可以选择不同的距离计算公式)进行模糊聚类的效果图。

(2) 在选择阈值过小情况下的聚类效果

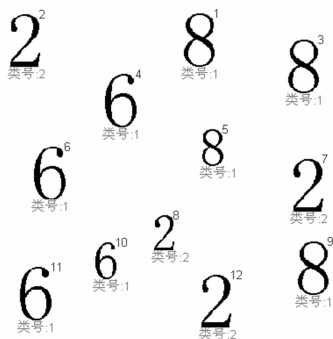
当用户给定的阈值过小时, 如图 10-9(a) 所示, 则会减少分类数目, 根据如图 10-9(b) 所示的模糊矩阵对照表, 将不同的样品归入同一类, 如图 10-9(c) 所示。



(a) 输入阈值过小

行列	2	3	4	5	6	7	8	9	10	11	12
1	1	0	1	1	1	1	0	0	1	1	0
2	0	1	0	0	0	0	1	1	0	0	1
3	1	0	1	1	1	1	0	0	1	1	0
4	1	0	1	1	1	1	0	0	1	1	0
5	1	0	1	1	1	1	0	0	1	1	0
6	1	0	1	1	1	1	0	0	1	1	0
7	0	1	0	0	0	0	1	1	0	0	1
8	0	1	0	0	0	0	1	1	0	0	1
9	1	0	1	1	1	1	0	0	1	1	0
10	1	0	1	1	1	1	0	0	1	1	0
11	1	0	1	1	1	1	0	0	1	1	0
12	0	1	0	0	0	0	1	1	0	0	1

(b) 模糊矩阵对照表



(c) 聚类结果, 不同类样品被归入同一类

图 10-9 阈值过小时的聚类结果

(3) 在选择阈值过大情况下的聚类效果

如果用户给定的阈值过大, 如图 10-10(a) 所示, 则会增大聚类数目, 根据如图 10-10(b) 所示的模糊矩阵对照表, 将本来属于同一类的样品分到不同的类中, 如图 10-10(c) 所示。

(4) 应用于图形情况下的聚类效果

模糊聚类法同样能应用于图形的聚类中, 如图 10-11(a) 所示。选择一种距离模式如图 10-11(b) 所示, 根据该距离模式产生如图 10-11(c) 所示的模糊矩阵, 给定阈值, 如图 10-11(d) 所示。输出聚类前后的模糊矩阵对照表, 如图 10-11(e) 所示, 输出模糊聚类的结果, 如图 10-11(f) 所示。由图可见, 模糊聚类法效果令人满意。

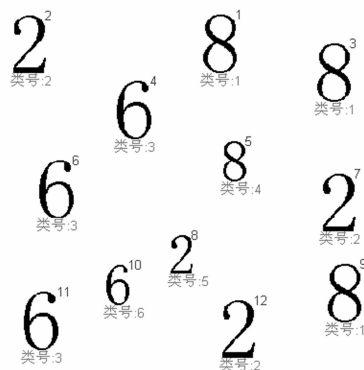


(a) 输入阈值过大

Figure 1: 每行中“1”对应的列为同一类

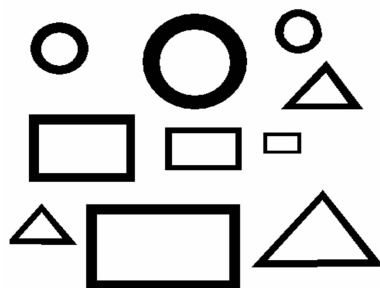
行/列	2	3	4	5	6	7	8	9	10	11	12
1	1	0	1	0	0	0	0	0	1	0	0
2	0	1	0	0	0	0	1	0	0	0	1
3	1	0	1	0	0	0	0	0	1	0	0
4	0	0	0	1	0	1	0	0	0	0	1
5	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	1	0	1	0	0	0	0	1
7	0	1	0	0	0	0	1	0	0	0	1
8	0	0	0	0	0	0	0	1	0	0	0
9	1	0	1	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	1	0	1	0	0	0	1	0
12	0	1	0	0	0	1	0	0	0	0	1

(b) 模糊矩阵对照表



(c) 聚类结果, 同类样品被归入不同类

图 10-10 阈值过大时的聚类结果



(a) 图形原文件



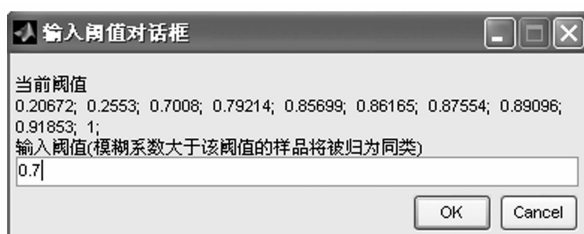
(b) 输入聚类模式

Figure 1: 模糊系数矩阵 (Fuzzy Coefficient Matrix)

行/列	1	2	3	4	5	6	7	8	9	10
1	1	0.89096	0.85699	0.20672	0.20672	0.20672	0.20672	0.20672	0.20672	0.20672
2	0.89096	1	0.85699	0.20672	0.20672	0.20672	0.20672	0.20672	0.20672	0.20672
3	0.85699	0.85699	1	0.20672	0.20672	0.20672	0.20672	0.20672	0.20672	0.20672
4	0.20672	0.20672	0.20672	1	0.2553	0.2553	0.2553	0.7008	0.79214	0.2553
5	0.20672	0.20672	0.20672	0.2553	1	0.91853	0.87554	0.2553	0.2553	0.86165
6	0.20672	0.20672	0.20672	0.2553	0.91853	1	0.87554	0.2553	0.2553	0.86165
7	0.20672	0.20672	0.20672	0.2553	0.87554	0.87554	1	0.2553	0.2553	0.86165
8	0.20672	0.20672	0.20672	0.7008	0.2553	0.2553	0.2553	1	0.7008	0.2553
9	0.20672	0.20672	0.20672	0.79214	0.2553	0.2553	0.2553	0.7008	1	0.2553
10	0.20672	0.20672	0.20672	0.2553	0.86165	0.86165	0.86165	0.2553	0.2553	1

(c) 输出该距离模式下的模糊矩阵

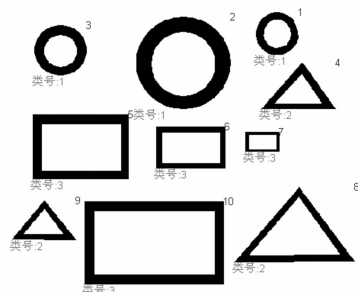
图 10-11 模糊聚类应用于图形聚类



(d) 输入模糊系数的阈值

行/列	2	3	4	5	6	7	8	9	10
1	1	1	1	0	0	0	0	0	0
2	1	1	1	0	0	0	0	0	0
3	1	1	1	0	0	0	0	0	0
4	0	0	0	1	0	0	0	1	1
5	0	0	0	0	1	1	1	0	0
6	0	0	0	0	1	1	1	0	0
7	0	0	0	0	1	1	1	0	0
8	0	0	0	1	0	0	0	1	1
9	0	0	0	1	0	0	0	1	1
10	0	0	0	0	1	1	1	0	0

(e) 输出聚类前后的模糊矩阵对照表



(f) 模糊聚类结果

图 10-11 模糊聚类应用于图形聚类(续)

本章小结

本章介绍了模糊集的基本概念,包括隶属函数和模糊子集的定义,介绍了模糊集运算及其性质,还介绍了模糊关系及模糊集在模式识别中的应用,详细介绍了模糊集理论应用于聚类问题的方法和步骤。

习题 10

1. 模糊集合与普通集合有什么区别?
2. 什么是隶属度函数?它和模糊子集有什么关系?
3. 叙述模糊集运算的性质。
4. 如何将一个普通模糊矩阵转化成等价矩阵?
5. 叙述将模糊集理论应用于聚类问题的步骤。

第 11 章 禁忌搜索算法聚类分析

11.1 禁忌搜索算法的基本原理

工程领域内存在大量的优化问题,对于优化算法的研究一直是计算机领域内的一个热点问题。优化算法主要分为全局优化算法和局部邻域搜索算法。全局优化算法不依赖问题的性质,按一定规则搜索解空间,直到搜索到近似最优解或最优解,属于智能随机算法,其代表有遗传算法、模拟退火算法、粒子群算法等。

局部邻域搜索是基于贪婪思想持续地在当前解的邻域中进行搜索,通常可描述为:从一个初始解出发,利用邻域函数持续地在当前解的邻域中搜索比它好的解,若能够找到如此的解,就以其成为新的当前解,然后重复上述过程;否则结束搜索过程,并以当前解作为最终解。局部搜索算法性能依赖于邻域结构和初始解,若邻域函数设计不当或初值选取不合适,则算法最终的性能会很差。局部邻域搜索算法依赖对问题性质的认识,易陷入局部极小而无法保证全局优化性,但是该算法属于启发式搜索,容易理解,通用且易实现。同时,贪婪思想无疑将使算法丧失全局优化能力,也即算法在搜索过程中无法避免陷入局部极小。因此,若不在搜索策略上进行改进,实现全局优化,局部搜索算法采用的邻域函数必须是“完全的”,即邻域函数将导致解的完全枚举。而这在大多数情况下无法实现,且穷举的方法对于大规模问题在搜索时间上是不允许的。为了实现全局优化,可尝试的途径有:扩大邻域搜索结构、多点并行搜索,如进化计算、变结构邻域搜索(Mladenovic et al., 1997)等。

禁忌搜索(Tabu Search 或 Tabu Search, TS)算法是对局部邻域搜索的一种扩展,搜索过程中采用禁忌准则,即不考虑处于禁忌状态的解,标记对应已搜索的局部最优解的一些对象,在进一步的迭代搜索中尽量避开这些对象(而不是绝对禁止循环),避免迂回搜索,从而保证对不同的有效搜索途径的探索,是一种局部极小突跳的全局逐步寻优算法。最早由 Glover (1986)提出。TS 算法是对人类智力过程的一种模拟,人工智能的一种体现。TS 算法在函数全局优化、组合优化、生产调度、机器学习等领域取得了很大的成功。

1. 禁忌搜索算法基本原理

禁忌搜索算法的基本思想是:给定一个初始解(随机的),作为当前最优解,给定一个状态“best so far”,作为全局最优解。给定初始解的一个邻域,然后在此初始解的邻域中确定若干解作为算法的候选解;利用适配值函数评价这些候选解,选出最佳候选解;若最佳候选解所对应的目标值优于“best so far”状态,则忽视它的禁忌特性,并且用这个最佳候选解替代当前解和“best so far”状态,并将相应的解加入到禁忌表中,同时修改禁忌表中各个解的任期;若候选解达不到以上条件,则在候选解里面选择非禁忌的最佳状态作为新的当前解,并且不管它与当前解的优劣,将相应的解加入到禁忌表中,同时修改禁忌表中各对象的任期;最后,重复上述搜索过程,直至满足停止准则。

简单的禁忌搜索算法是在邻域搜索的基础上,通过引入一个灵活的存储结构和相应的禁忌准则来避免迂回搜索,存储结构存放禁忌表,记录已经历的一些禁忌操作,并利用藐视准则来奖励赦免一些被禁忌的优良状态,进而保证多样化的有效探索以最终实现全局优化。

简单禁忌搜索的算法步骤可描述如下:

- (1) 给定算法参数,包括候选解(candidate)的选取个数,禁忌表长度(tabu length)等。
- (2) 随机产生初始解 x ,置禁忌表为空。
- (3) 判断算法终止条件是否满足,若是,则结束算法并输出优化结果;否则,继续以下步骤。
- (4) 利用当前解的邻域函数产生其所有(或若干)邻域解,并从中确定若干候选解。
- (5) 对候选解判断藐视准则是否满足,若成立,则用满足藐视准则的最佳状态 y 替代 x 成为新的当前解,并用与 y 对应的禁忌对象替换最早进入禁忌表的禁忌对象,同时用 y 替换“best so far”状态,然后转步骤(7);否则,继续以下步骤。
- (6) 判断候选解对应的各对象的禁忌属性,选择候选解集中非禁忌对象对应的最佳状态为新的当前解,同时用与之对应的禁忌对象替换最早进入禁忌表的禁忌对象元素。
- (7) 转步骤(3)。

算法法可用图 11-1 所示的流程图更为直观地描述。

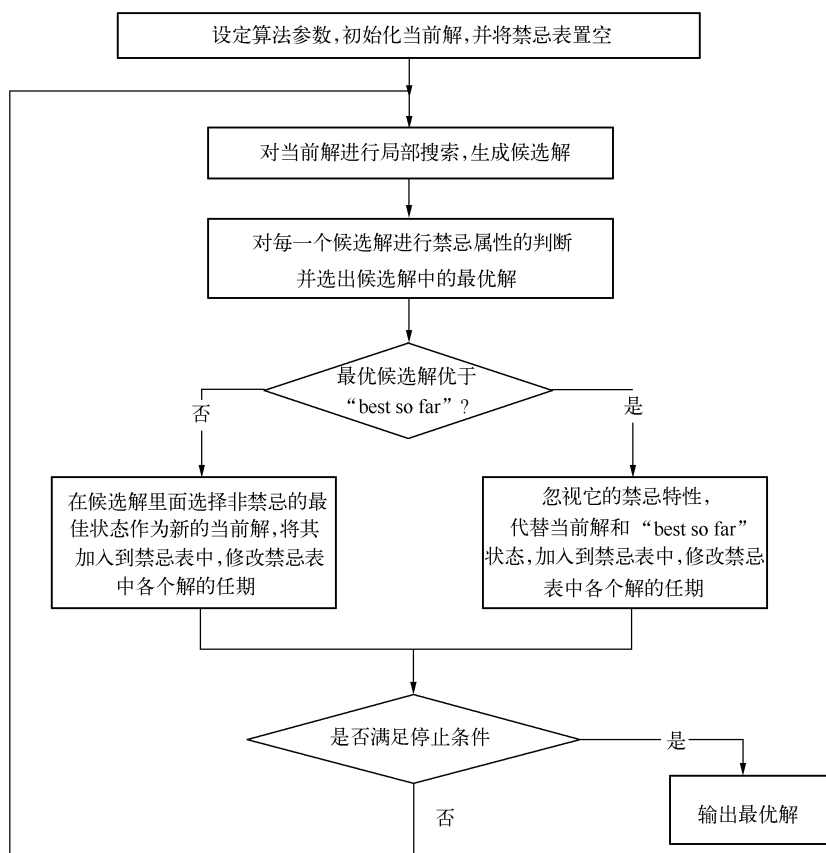


图 11-1 禁忌搜索算法基本流程

需要指出的是,上述算法仅是一种简单的禁忌搜索框架,邻域函数、禁忌对象、禁忌表和藐视准则,构成了禁忌搜索算法的关键。对各关键环节复杂和多样化的设计则可构造出各种禁忌搜索算法。

2. 禁忌搜索算法的优缺点

由于禁忌搜索算法具有灵活的记忆功能和藐视准则,并且在搜索过程中可以接受劣解,所以具有较强的“爬山”能力,搜索时能够跳出局部最优解,转向解空间的其他区域,从而增加获得更好的全局最优解的概率,所以禁忌搜索算法是一种局部搜索能力很强的全局迭代寻优算法。

区别于传统的优化算法,禁忌搜索算法的主要特点是搜索过程中可以接受劣解,因此具有较强的“爬山”能力,新解不是在当前解的邻域中随机产生,而是优于“best so far”的解,或是非禁忌的最佳解,因此选取优良解的概率远远大于其他解。

迄今为止,尽管禁忌搜索算法在许多领域得到了成功应用。禁忌搜索也有明显不足,对初始解的依赖性较强,好的初始解有助于搜索很快达到最优解,而较坏的初始解往往会使搜索很难或不能达到最优解,因此有先验知识指导下的初始解更容易让算法找到最优解。并且迭代搜索过程是串行的,仅是单一状态的移动,而非并行搜索。

11.2 禁忌搜索的关键参数和相关操作

这里主要从实现技术上介绍禁忌搜索算法最基本的操作和参数的常用设计原则和方法,包括适配值函数、邻域函数、禁忌表和禁忌对象的设置、禁忌长度、候选解集、藐视准则、禁忌频率和终止准则等。

1. 适配值函数

类似于遗传算法,禁忌搜索的适配值函数也是用于对搜索状态的评价,进而结合禁忌准则和藐视准则来选取新的当前状态。显然,目标函数直接作为适配值函数是比较容易理解的做法。当然,目标函数的任何变形都可作为适配值函数。若目标函数的计算比较困难或耗时较多,如一些复杂工业过程的目标函数值需要一次仿真才能获得,此时可以将反映问题目标的某些特征值作为适配值,进而改善算法的时间性能。当然,选取何种特征值要视具体问题而定,但必须保证特征值的最佳性与目标函数的最优性一致。

2. 邻域函数

邻域函数沿用局部邻域搜索的思想,用于实现邻域搜索。邻域函数是优化中的一个重要概念,其作用就是指导如何由一个(组)解来产生一个(组)新的解。邻域函数的设计往往依赖于问题的特性和解的表达方式,应结合具体问题进行分析。

3. 候选解集

候选解集则通常是当前状态的邻域解集的一个子集。候选解集的大小是影响 TS 算法性能的关键参数。候选解通常在当前状态的邻域中择优选取,但选取过多将造成较大的计算量,而选取过少则容易造成早熟收敛。然而,要做到整个邻域的择优往往需要大量的计算,因此可以确定性或随机性地在部分邻域解中选取候选解,具体数据大小则可视问题特性和对算法的

要求而定。

4. 禁忌准则

标记对应已搜索的局部最优解的一些对象,将这些已经搜索过的对象设定为禁忌状态,在进一步的迭代搜索中不考虑处于禁忌状态的解,尽量避开这些对象(而不是绝对禁止循环),避免迂回搜索,从而保证对不同的有效搜索途径的探索,是一种局部极小突跳的全局逐步寻优算法。

5. 藐视准则

藐视准则是对优良状态的奖励,它是对禁忌策略的一种放松。在禁忌搜索算法中,若存在优于“best so far”状态的禁忌候选解,则将最优禁忌候选解从禁忌表解禁;或者可能会出现候选解全部被禁忌,此时藐视准则将使最优禁忌候选解从禁忌表解禁,以实现更高效的优化性能。在此给出藐视准则的几种常用方式。

(1) 基于适配值的准则。若某个禁忌候选解的适配值优于“best so far”状态,则解禁此候选解为当前状态和新的“best so far”状态;也可以将搜索空间分成若干个子区域,若某个禁忌候选解的适配值优于它所在区域的“best so far”状态,则解禁此候选解为当前状态和相应区域的新“best so far”状态。该准则可直观理解为算法搜索到了一个更好的解。

(2) 基于最小错误的准则。若候选解均被禁忌,且不存在优于“best so far”状态的候选解,则对候选解中最优的候选解进行解禁,以继续搜索。该准则可直观理解为对算法死锁的简单处理。

(3) 基于搜索方向的准则。若禁忌对象上次使得适配值有所改善,被禁忌加入禁忌表,但是目前该禁忌对象对应的候选解的适配值优于当前解,则对该禁忌对象解禁。该准则可直观理解为算法正按有效的搜索途径进行。

(4) 基于影响力的准则。在搜索过程中不同对象的变化对适配值的影响有所不同,有的很大,有的较小,而这种影响力可作为一种属性与禁忌长度和适配值来共同构造藐视准则。直观的理解是,解禁一个影响力大的禁忌对象,有助于在以后的搜索中得到更好的解。需要指出的是,影响力仅是一个标量指标,可以表征适配值的下降,也可以表征适配值的上升。譬如,若候选解均差于“best so far”状态,而某个禁忌对象的影响力指标很高,且很快将被解禁,则立刻解禁该对象以期待更好的状态。显然,这种准则需要引入一个标定影响力大小的度量和一个与禁忌任期相关的闭值,无疑增加了算法操作的复杂性。同时,这些指标最好是动态变化的,以适应搜索进程和性能的变化。

6. 禁忌表和禁忌对象的设置

禁忌对象就是被置入禁忌表中的那些变化元素,而禁忌的目的则是为了尽量避免迂回搜索而多探索一些有效的搜索途径。禁忌表和禁忌对象的设置体现了算法避免迂回搜索的特点。禁忌对象通常可选取状态本身或状态分量或适配值的变化等。以状态本身或其变化作为禁忌对象是最为简单、最容易理解的途径。具体而言,当状态由 x 变化至状态 y 时,将状态 y (或 x 到 y 的变化) 视为禁忌对象,从而在一定条件下禁止了 y (或 x 到 y 的变化) 的再度出现。

7. 禁忌长度

所谓禁忌长度,即禁忌对象在不考虑藐视准则情况下不允许被选取的最大次数(也可视

为对象在禁忌表中的任期),对象只有当其任期为 0 时才被解禁。在算法的构造和计算过程中,一方面要求计算量和存储量尽量少,这就要求禁忌长度尽量小;但是,禁忌长度过短将造成搜索的循环。禁忌长度的大小是影响 TS 算法性能的关键参数。禁忌长度的选取与问题特性、研究者的经验有关,它决定了算法的计算复杂性。

禁忌长度可以是定长不变的,如将禁忌长度固定为某个数,或者固定为与问题规模相关的一个量。另一方面,禁忌长度也可以是动态变化的,如根据搜索性能和问题特性设定禁忌长度的变化区间 $[t_{\min}, t_{\max}]$,而禁忌长度则可按某种原则或公式在其区间内变化。当然,禁忌长度的区间大小也可随搜索性能的变化而动态变化。

当算法的性能动态下降较大时,说明当前解附近极小解形成的“波谷”较深,算法当前的搜索能力比较强,从而可设置较大的禁忌长度来延续当前的搜索行为,并避免陷入局部极小。研究表明,禁忌长度的动态设置方式比静态方式具有更好的性能和鲁棒性,而更为合理高效的设置方式还有待进一步研究。

8. 禁忌频率

记忆禁忌频率(或次数)是对禁忌属性的一种补充,可放宽选择决策对象的范围。譬如,如果某个适配值频繁出现,则可以推测算法陷入某种循环或某个极小点,或者说现有算法参数难以有助于发掘更好的状态,进而应当对算法结构或参数进行修改。在实际求解时,可以根据问题和算法的需要,记忆某个状态出现的频率,也可以是某些对换对象或适配值等出现的信息,而这些信息可以是静态的,或者是动态的。

静态的频率信息主要包括状态、适配值或对换等对象在优化过程中出现的频率,其计算相对比较简单,如对象在计算中出现的次数,出现次数与总迭代步数的比,某两个状态间循环的次数等。显然,这些信息有助于了解某些对象的特性,以及相应循环出现的次数等。

动态的频率信息主要记录从某些状态、适配值或对换等对象转移到另一些状态、适配值或对换等对象的变化趋势,如记录某个状态序列的变化。显然,对动态频率信息的记录比较复杂,而它所提供的信息量也较多。常用的方法如下:

(1) 记录某个序列的长度,即序列中的元素个数,而在记录某些关键点的序列中,可以按这些关键点的序列长度的变化来进行计算。

(2) 记录由序列中的某个元素出发后再回到该元素的迭代次数。

(3) 记录某个序列的平均适配值,或者是相应各元素的适配值的变化。

(4) 记录某个序列出现的频率等。

上述频率信息有助于加强禁忌搜索的能力和效率,并且有助于对禁忌搜索算法参数的控制,或者可基于此对相应的对象实施惩罚。譬如,若某个对象频繁出现,则可以增加禁忌长度来避免循环;若某个序列的适配值变化较小,则可以增加对该序列所有对象的禁忌长度,反之则缩小禁忌长度;若最佳适配值长时间维持下去,则可以终止搜索进程而认为该适配值已是最优值。

9. 终止准则

与模拟退火、遗传算法一样,禁忌搜索也需要一个终止准则来结束算法的搜索进程,而严格实现理论上的收敛条件,即在禁忌长度充分大的条件下实现状态空间的遍历,这显然是不切合实际的,因此实际设计算法时通常采用近似的收敛准则。常用方法如下:

- (1) 给定最大迭代步数。此方法简单易操作,但难以保证优化质量。
- (2) 设定某个对象的最大禁忌频率。即若某个状态、适配值或对换等对象的禁忌频率超过某一阈值,则终止算法,其中也包括最佳适配值连续若干次迭代并保持不变的情况。

11.3 基于禁忌搜索算法的聚类分析

1. 问题的提出

本节以图像中不同物体聚类分析为例,介绍用禁忌搜索算法解决聚类问题的实现方法。

一幅图像中含有多个物体,在图像中进行聚类分析需要对不同的物体分割标识,如图 11-2 所示,手写了(12322 13132)共 10 个待分类样品,要分成 3 类,如何让计算机自动将这 10 个物体归类呢?禁忌搜索算法在解决这种聚类问题上表现非常出色,它不仅运算速度快,而且准确率高。

2. 解的编码形式

对图 11-2 所示的 10 个物体进行编号,样品编号如图 11-3 所示,在每个样品的右上角,不同的样品编号不同,而且编号始终固定。

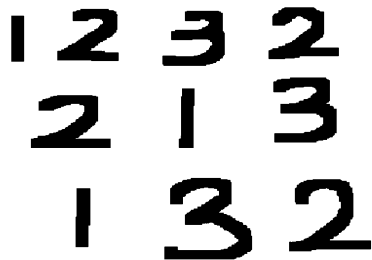


图 11-2 待聚的样品数字

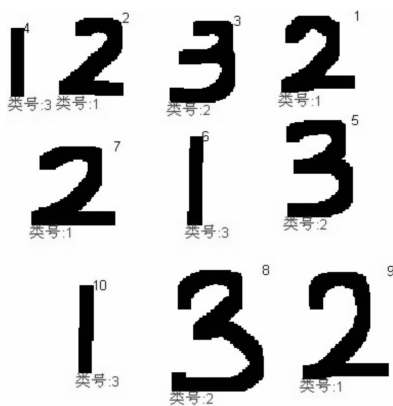


图 11-3 待测样品的编号

采用符号编码,位串长度 L 取 10 位,分类号代表样品所属的类号(1~3),样品编号是固定的,也就是说某个样品在每个解中的位置是固定的,而每个样品所属的类别随时在变化。如果编号为 n ,则其对应第 n 个样品,而第 n 个位所指向的值代表第 n 个样品的归属类号。

每个解包含一种分类方案。设初始解的编码为:(2,3,2,1,3,3,2,3,1,3),这时处于假设分类情况,不是最优解,其含义为:第 1、3、7 个样品被分到第 2 类;第 2、5、6、8、10 个样品被分到第 3 类;第 4、9 个样品被分到第 1 类;如表 11-1 所示。

表 11-1 初始解

样品值	(2)	(2)	(3)	(1)	(3)	(1)	(2)	(3)	(2)	(1)
分类号	2	3	2	1	3	3	2	3	1	3
样品编号	1	2	3	4	5	6	7	8	9	10

经过禁忌搜索算法找到的最优解如图 11-3 所示,类号在每个样品的左下角。禁忌搜索算法找到的最优解编码如表 11-2 所示。通过样品值与基因值比较对照,就会发现相同的数据被归为一类,分到相同的类号,而且全部正确。

表 11-2 禁忌搜索算法最终找到的最优解

样品值	(2)	(2)	(3)	(1)	(3)	(1)	(2)	(3)	(2)	(1)
分类号	1	1	2	3	2	3	1	2	1	3
样品编号	1	2	3	4	5	6	7	8	9	10

3. 设定适配值函数

适配值函数 CalObjfitness() 结果为评估值,代表每个解优劣的程度。初始解中的评估值 m_solution. fitness 计算步骤如下:

① 通过人工干预获得聚类类别总数,centerNum 为聚类类别总数($2 \leq \text{centerNum} \leq N - 1$, N 是总的样品个数)。

② 找出解中相同类号的样品, $X^{(i)}$ 表示属于第 i 个类的样品。

③ 统计每一个类的样品个数 n , n_i 是第 i 个类别的个数,样品总数为: $N = \sum_{i=1}^{\text{centerNum}} n_i$ 。

④ 计算同一个类的中心 C , C_i 是第 i 个类中心,

$$C_i = \frac{1}{n_i} \sum_{k=1}^{n_i} X_k^{(i)}, (i = 1, 2, \dots, \text{centerNum});$$

⑤ 同一个类内计算每一个样品到中心的距离,并将它们累加求和。

采用 K -均值模型为聚类模型, K -均值的理论已经在第 8 章介绍过。计算公式如下:

$$D_i = \sum_{j=1}^{n_i} \|X_j^{(i)} - C_i\|^2 \quad (11-1)$$

显然,当聚类类别总数的数目 $\text{centerNum} = N$ 时,累加和 $\sum D_i$ 为 0。因此,当聚类数目 centerNum 不定时,必须对目标函数进行修正。实际上,式(11-1)仅为类内距离之和,因此,可以使用类内距离与类间距离之和作为目标函数,即:

$$D = \min \left[w * \sum_{i=1}^{\text{centerNum}} \sum_{j=1}^{n_i} \|X_j^{(i)} - C_i\|^2 + \sum_{i=1}^{\text{centerNum}} \sum_{j=i}^{\text{centerNum}} \|C_i - C_j\|^2 \right] \quad (11-2)$$

式中, w 是权重,反映决策者的偏爱;当 w 变大时,聚类总数数目将变大,反之聚类总数数目将变小。

⑥ 将不同类计算出的 D_i 求和赋给 m_solution. fitness,以 m_solution. fitness 作为评估值。

$$\text{m_solution. fitness} = \frac{1}{\sum_{i=1}^{\text{centerNum}} \sum_{j=1}^{n_i} \|X_j^{(i)} - C_i\|^2} = \frac{1}{\sum_{i=1}^{\text{centerNum}} D_i} \quad (11-3)$$

m_solution. fitness 越大,说明这种分类方法的误差越小,该解作为最优解的机率也就越大。

4. 邻域函数和局部搜索

由于局部搜索算法是基于贪婪思想利用邻域函数进行搜索的,对邻域函数和初始解有很

大的依赖性。因此,必须设置合理的邻域函数。即要求是“完全的”,同时也要考虑到时间和效率的因素。因此,这里采用位变异的方法进行局部搜索,来获得候选解。需要注意的是,如果变异概率系数过小,则解的变化波动性小,说明算法搜索能力比较弱,容易陷入局部极小。反之,如果变异概率系数过大,则解的变化范围大,算法易陷入盲目搜索状态,不利于算法收敛。更为合理高效的设置方式还需要多次调整。

5. 禁忌表

以解的形式本身作为禁忌对象,构造一个禁忌表,每一行存着处于禁忌状态的解。禁忌长度的大小是影响 TS 算法性能的关键参数。这里禁忌长度采用固定长度。当有一个新解需要被禁忌时,用该解代替最早加入禁忌表的解,即一个最早记忆的解被遗忘,从禁忌表中解禁出来。

6. 藐视准则

如果最优候选解的评估值优于全局最优解,且最优候选解处于禁忌状态,则采用藐视准则,将该解代替当前解和全局最优解。反之,如果所有候选解都处于禁忌状态,则用最优候选解代替当前解。

7. 实现步骤

① 设置相关参数。

从对话框中输入各参数,包括类中心数(centerNum),最大迭代次数(MaxIter = 30),选取的候选解的个数(CandidateNum = 40),禁忌表长度(TabuLength = 60)。

② 初始化

在解域中随机找寻一点,即随机初始化各样品的类号。同时构造禁忌表,表长为 TabuLength,并将其置空。

③ 计算初始解对应的评估值 fitness。

④ 在当前解的邻域内找 CandidateNum 个候选解。

设定候选解集总大小 CandidateNum,设定变异概率系数 $pm = 0.25$ 。重复 CandidateNum 次,产生候选解。对于当前解的每一位,生成一个 $0 \sim 1$ 之间的随机数 rand。当该随机数小于变异概率系数时,让该位随机改变成 $1 \sim \text{centerNum}$ 之间的一个数。

⑤ 判断每个候选解的禁忌状态。

对每一个候选解,循环禁忌表中每一行,与该候选解进行比较。如果存在相同的情况,即该候选解已被记录在表中,处于禁忌状态,将该候选解的禁忌标识 bTabu 置为 1,否则置为 0。

⑥ 从候选解中依据适配度值进行排序,找到最优候选解。

⑦ 如果该解的适配度值高于全局最优解,则忽视它的禁忌特性,将该候选解代替当前解和全局最优解,并加入禁忌表,代替最早进入禁忌表的解。

否则,根据候选解的禁忌状态,若存在非禁忌状态下的候选解,用非禁忌状态下的最优候选解代替当前解,并将其加入禁忌表,代替最早进入禁忌表的解;若不存在非禁忌状态下的候选解,即所有候选解都处于禁忌状态,则用最优候选解代替当前解。

⑧ 比较当前解与全局最优解的评估值,更新全局最优解。

⑨ 检查是否达到停止条件(即达到最大迭代次数)。如果是,则跳出迭代,输出全局最优解。否则转到步骤④,继续迭代。

基于禁忌搜索算法的聚类分析流程图如图 11-4 所示。

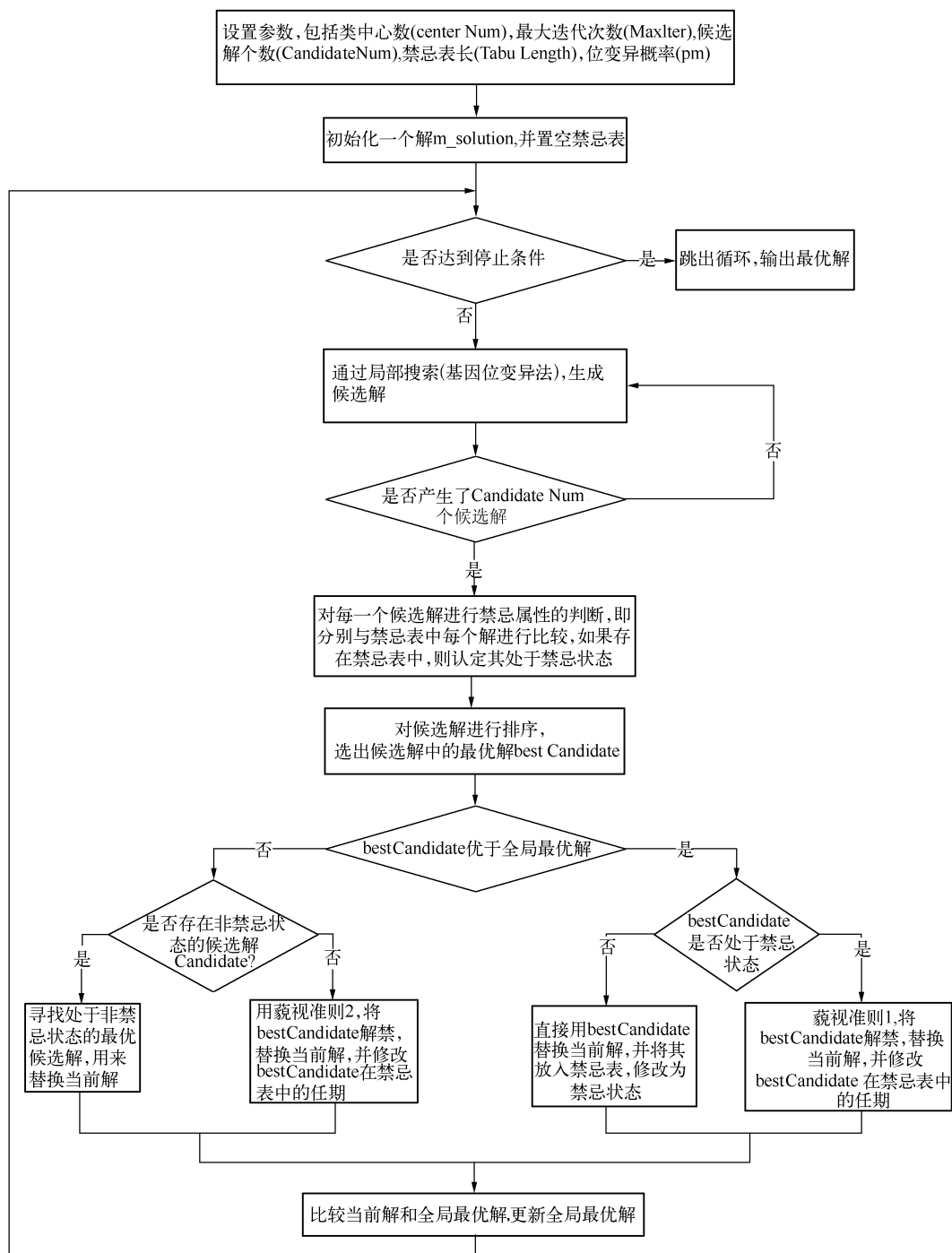


图 11-4 基于禁忌搜索算法的聚类分析流程图

8. 编程代码

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 函数名称:C_TS( )
% 参数:m_pattern:样品特征库;patternNum:样品数目
% 返回值:m_pattern:样品特征库
% 函数功能:按照禁忌搜索算法对全体样品进行聚类
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ m_pattern ] = C_TS( m_pattern,patternNum )
    disType = DisSelDlg( );% 获得距离计算类型
    [ centerNum MaxIter CandidateNum TabuLength pm ] = InputParameterDlg( );% 获得参数
% 初始化
    m_solution.string = ceil( rand( 1,patternNum) * centerNum );% 当前解位串
    m_solution.fitness = 0;% 适应度
    m_solution.index = 1;
% 计算适应度值
    m_solution = CalOneFitness( m_solution,patternNum,centerNum,m_pattern,disType );
% 初始化全局最优个体为当前解
    cBest = m_solution;
% 初始化候选个体
    for i = 1:CandidateNum
        Candidate(i).string = ceil( rand( 1,patternNum) * centerNum );% 个体位串
        Candidate(i).fitness = 0;% 适应度
        Candidate(i).bTabu = 0;
    end
% 禁忌表
    TabuList = zeros( TabuLength ,patternNum );
% 迭代计算
    for iter = 2:MaxIter
        % 生成候选解
        Candidate = SelectCandidate( m_solution,Candidate,CandidateNum,patternNum,centerNum,pm );
        % 计算适应度值
        for k = 1:CandidateNum
            Candidate( k ) = CalOneFitness( Candidate( k ), patternNum, centerNum, m_pattern, dis-
Type );
        end
        % 依据适配值对候选解进行排序
        for k = 1:CandidateNum - 1
            for l = k + 1:CandidateNum
                if Candidate( k ).fitness < Candidate(l).fitness
                    temp = Candidate( k );
                    Candidate( k ) = Candidate( l );
                    Candidate( l ) = temp;
                end
            end
        end
    end
end

```

```

        end
    end
end
% 禁忌属性判断,若在禁忌表中,则让 bTabu 属性值为 1,否则为 0。
for k = 1:CandidateNum
    for l = 1:TabuLength
        if Candidate(k).string == TabuList(l,:)
            Candidate(k).bTabu = 1;
            break;
        else
            Candidate(k).bTabu = 0;
        end
    end
end
% 判断所有候选解是否都在禁忌表中
bAllTabu = 1;
for k = 1:CandidateNum
    if Candidate(k).bTabu == 0
        bAllTabu = 0;
        break;
    end
end
bestCandidate = Candidate(1);
% 如果该最优候选解优于全局最优解,则忽视它的禁忌特性,则替换当前解和全局最优解,并将该状态压入禁忌表
if bestCandidate.fitness > cBest.fitness
    m_solution = bestCandidate;
    for l = 1:TabuLength - 1
        TabuList(l,:) = TabuList(l+1,:);
    end
    TabuList(TabuLength,:) = bestCandidate.string;
else
    % 判断是否所有候选解都处于禁忌状态
    if bAllTabu == 0 % 如果不是,则用非禁忌的最优候选解代替当前解
        for k = 1:CandidateNum
            if Candidate(k).bTabu == 0
                m_solution = Candidate(k);
                for l = 1:TabuLength - 1
                    TabuList(l,:) = TabuList(l+1,:);
                end
                TabuList(TabuLength,:) = Candidate(k).string;
                break;
            end
        end
    end
end

```

```

        end
    else % 如果是,则用最优候选解代替当前解
        m_solution = bestCandidate;
        for l = 1:TabuLength - 1
            TabuList(l,:) = TabuList(l+1,:);
        end
        TabuList(TabuLength,:) = bestCandidate.string;
    end
end
end
if m_solution.fitness > cBest.fitness % 更新全局最优解
    cBest = m_solution;
    cBest.index = iter;
end
end
for i = 1:patternNum
    m_pattern(i).category = cBest.string(1,i);
end
% 显示结果
str = ['最优解出现在第 'num2str(cBest.index)'代'];
msgbox(str,'modal');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 函数名称: CalObjfitness()
% 参数: m_solution: 要计算评估值的解结构; patternNum: 样品数目;
%       centerNum: 类中心数; m_pattern: 样品特征库;
%       disType: 距离类型
% 返回值: m_solution: 要计算评估值的解结构;
% 函数功能: 计算解的评估值
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ m_solution ] = CalOneFitness( m_solution, patternNum, centerNum, m_pattern, disType )
    global Nwidth;
    for j = 1:centerNum % 初始化聚类中心
        m_center(j).index = i;
        m_center(j).feature = zeros( Nwidth, Nwidth );
        m_center(j).patternNum = 0;
    end
    % 计算聚类中心
    for j = 1:patternNum
        m_center( m_solution.string(1,j) ).feature = m_center( m_solution.string(1,j) ).feature + m_
pattern(j).feature;
        m_center( m_solution.string(1,j) ).patternNum = m_center( m_solution.string(1,j) ).pat-
ternNum + 1;
    end
    d = 0;

```

```

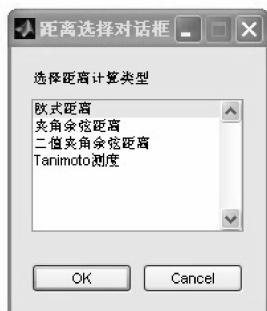
for j = 1:centerNum
    if(m_center(j). patternNum ~ =0)
        m_center(j). feature = m_center(j). feature/m_center(j). patternNum;
    else
        d = d + 1;
    end
end
m_solution. fitness = 0;
% 计算个体评估值
for j = 1:patternNum
    m_solution. fitness = m_solution. fitness + GetDistance( m_center( m_solution. string( 1,j) ), m_pat-
tern(j) ,disType)^2;
end
m_solution. fitness = 1/( m_solution. fitness + d) ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 函数名称:SelectCandidate( )
% 参数:m_solution:当前解结构;Candidate:临域解结构;CandidateNum:选取候选解个数
%      patternNum:样品数目;centerNum:类中心数;
% 返回值:Candidate:候选解结构;
% 函数功能:从临域中选择候选解
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ Candidate ] = SelectCandidate( m_solution , Candidate , CandidateNum , patternNum , centerNum ,
pm)
    for i = 1: CandidateNum
        Candidate(i). string = m_solution. string;
        for j = 1: patternNum
            if rand < pm % 位变异概率
                Candidate(i). string( 1,j) = ceil( centerNum * rand) ;
            end
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 函数名称:InputParameterDlg( )
% 返回值:centerNum:聚类中心数;MaxIter:最大迭代次数;
%      CandidateNum:候选解个数;TabuLength:禁忌表长度
% 函数功能:用户输入参数
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ centerNum MaxIter CandidateNum TabuLength pm ] = InputParameterDlg( )
    str = {'聚类中心数:', '最大迭代次数 ', '候选解个数 ', '禁忌表规模 ', '位变异概率'} ;
    def = {' ', '30', '40', '60', '0. 25' } ;
    T = inputdlg( str, '参数输入对话框 ', 1, def) ;
    centerNum = str2num( T{ 1, 1 } ) ;
    MaxIter = str2num( T{ 2, 1 } ) ;

```

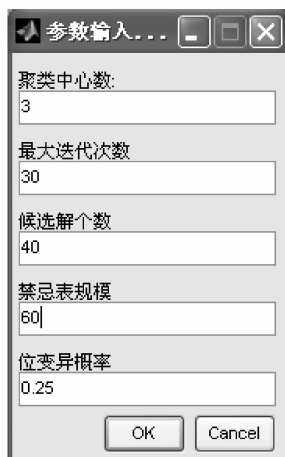
```
CandidateNum = str2num(T{3,1});  
TabuLength = str2num(T{4,1});  
pm = str2num(T{5,1});
```

9. 效果图

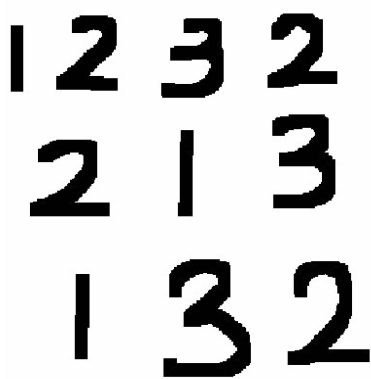
这里给读者提供一个基于手写数字聚类结果图,如图 11-5 所示。从结果上可以看出,禁忌搜索算法应用于聚类分析效果很好。



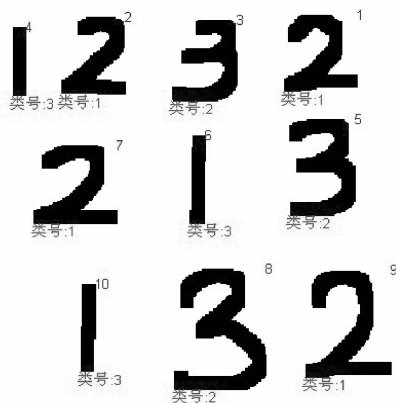
(a) 距离选择对话框



(b) 参数输入对话框



(c) 待聚类的样品



(d) 输出聚类结果



(e) 显示最优解出现在第几次迭代中

图 11-5 禁忌搜索算法应用于数字聚类分析

本章小结

求解聚类问题的算法很多。本章介绍了禁忌搜索算法基本原理,禁忌搜索算法的关键参数和相关操作,着重介绍了禁忌搜索算法用于聚类分析的实现方法和步骤。

本书主要介绍基本聚类算法、启发式方法和群体智能搜索三大类算法。第九章已经介绍了基本的聚类算法,其中典型的算法有:层次聚类算法, K 均值算法和迭代自组织的数据分析算法,这些算法采用点对点计算,不产生新解。基于模拟退火思想改进的 K 均值聚类算法和禁忌搜索算法属于启发式方法,是对局部邻域搜索扩展后的一种全局逐步寻优算法,在单个解基础上,产生新解,模拟退火算法每次产生一个新的候选解。而禁忌搜索算法在单个解的基础上产生多个新的候选解,模拟退火算法对新产生的候选解评价后,若优于当前解,则直接采用;而禁忌搜索算法采用禁忌准则,即不考虑处于禁忌状态的解,避免迂回搜索,从而保证对不同的有效搜索途径的探索。第 12、13、14 章介绍群体智能搜索算法,包括基于进化计算的遗传算法、蚁群算法和粒子群算法等,它们属于全局优化算法,采用全局分布随机产生多个初始解,利用进化算子或群体配合机制,使不同解之间产生信息交互,从而达到寻优目的。这些算法各有不同的特点,请读者仔细思考它们的寻优机制。

习题 11

1. 简述禁忌搜索算法的基本原理
2. 叙述基于模拟退火思想改进的 K 均值聚类算法和禁忌搜索算法的异同。
3. 叙述禁忌搜索算法在聚类问题中的实现方法。

第 12 章 遗传算法聚类分析

- 本章要点：
- ☑ 遗传算法的基本概念
 - ☑ 遗传算法的构成要素
 - ☑ 控制参数的选择
 - ☑ 基于遗传算法的聚类分析

12.1 遗传算法的基本原理

遗传算法(Genetic Algorithms ,GA)是一种新近发展起来的搜索最优解方法。它模拟生命进化机制,也就是说,模拟了自然选择和遗传进化中发生的繁殖、交配和突变现象,从任意一个初始种群出发,通过随机选择、交叉和变异操作,产生一群新的更适应环境的个体,使群体进化到搜索空间中越来越好的区域。这样一代一代不断繁殖、进化,最后收敛到一群最适应环境的个体上,求得问题的最优解。遗传算法对于复杂的优化问题无须建模和复杂运算,只要利用遗传算法的三种算子就能得到最优解。

经典遗传算法的一次进化过程示意图如图 12-1 所示,该图给出了第 n 代群体经过选择、交叉、变异生成第 $n + 1$ 代群体的过程。

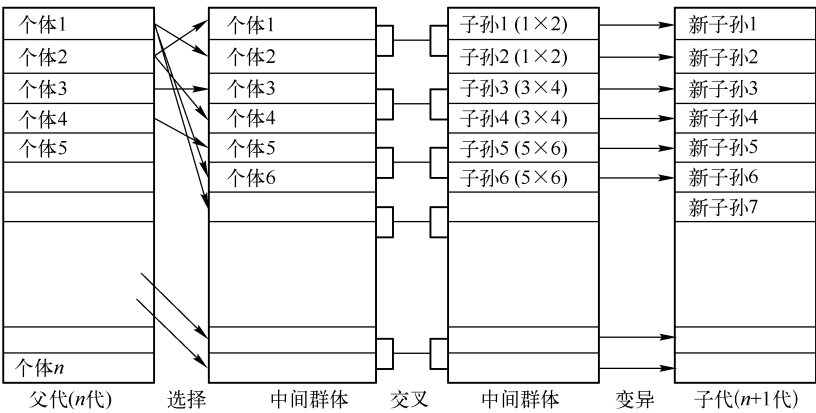


图 12-1 遗传算法的一次进化过程

1. 术语介绍

从图 12-1 可见,遗传算法涉及一些基本概念,下面对常见的术语进行解释。

- ① 个体(individual):GA 所处理的基本对象、结构。
- ② 群体(population):个体的集合。

- ③ 位串(bit string):个体的表示形式,对应于遗传学的染色体(chromosome)。
- ④ 基因(gene):位串中的元素,表示不同的特征,对应于生物学中的遗传物质单位,以 DNA 序列形式把遗传信息译成编码。
- ⑤ 基因位(locus):某一基因在染色体中的位置。
- ⑥ 等位基因(allele):表示基因的特征值,即相同基因位的基因取值。
- ⑦ 位串结构空间(bit string space):等位基因任意组合构成的位串集合,基因操作在位串结构空间进行,对应于遗传学中的基因型的集合。
- ⑧ 参数空间(parameters space):是指位串空间在物理系统中的映射,对应于遗传学中的表现型的集合。
- ⑨ 适应值(fitness):某一个体对于环境的适应程度,或者在环境压力下的生存能力,取决于遗传特性。
- ⑩ 选择(selection):在有限资源空间上的排他性竞争。
- ⑪ 交叉(crossover):一组位串或染色体上对应基因段的交换。
- ⑫ 变异(mutation):染色体水平上的基因变化,可以遗传给子代个体。
- 为了说明这些术语的含义,如图 12-2 所示为对个体的形象化表示。

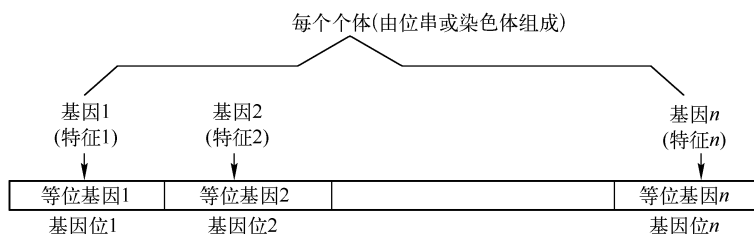


图 12-2 个体结构(由位串或染色体组成)

2. 遗传算法进行问题求解过程

遗传算法进行问题求解过程如下所述。

- ① 编码:遗传算法在求解之前,先将问题解空间的可行解表示成遗传空间的基因型串结构数据,串结构数据的不同组合构成了不同的可行解。
- ② 生成初始群体:随机产生 N 个初始串结构数据,每个串结构数据成为一个个体, N 个个体组成一个群体,遗传算法以该群体作为初始迭代点。
- ③ 适应度评估检测:根据实际标准计算个体的适应度,评判个体的优劣,即该个体所代表的可行解的优劣。
- ④ 选择:从当前群体中选择优良的(适应度高的)个体,使它们有机会被选中进入下一次迭代过程,舍弃适应度低的个体。体现了进化论的“适者生存”原则。
- ⑤ 交叉:遗传操作,下一代中间个体的信息来自父辈个体,体现了信息交换的原则。
- ⑥ 变异:随机选择中间群体中的某个个体,以变异概率 P_m 大小改变个体某位基因的值。变异为产生新个体提供了机会。

经典遗传算法流程图如图 12-3 所示,算法完全依靠三个遗传算子进行求解,当停止运算条件满足时,达到最大循环次数,同时最优个体不再进化。

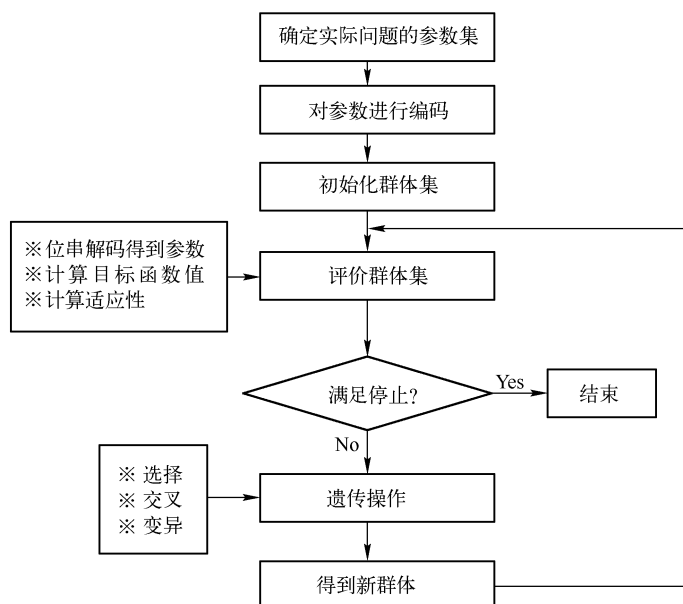


图 12-3 遗传算法流程图

3. 遗传算法的特点

遗传算法与其他优化算法相比,主要有以下几个特点:

① 遗传算法以问题参数的编码集为处理对象。传统的优化算法往往直接利用问题的参数的实际值本身来进行优化计算,通过调整参数的值找到最优解。但是遗传算法通过将参数编码,在求解问题的决定因素和控制参数的编码集上进行操作,因而不受函数限制条件(如导数的存在,连续性,单调性等)的约束,可以解决传统方法不能解决的问题。

② 遗传算法求解是从问题的解位串集开始搜索,而不是单个解开始搜索,搜索空间范围大,降低了陷入局部最优的可能性。

③ 遗传算法仅使用目标函数来进行搜索,不需要其他辅助信息。这使遗传算法大大扩大了应用范围,因为目标函数总是存在的,所以遗传算法可以应用在绝大部分问题中。

④ 遗传算法的三个遗传算子(选择、交叉、变异)是随机的。

⑤ 隐含的并行性。

⑥ 可扩展性,易于同其他技术结合。

12.2 遗传算法的构成要素

12.2.1 染色体的编码

所谓编码,就是指将问题的解空间转换成遗传算法所能处理的搜索空间。进行遗传算法求解前,必须对问题的解空间进行编码,以使它能够被遗传算法的算子操作。例如,用遗传算法进行模式识别,将样品标识后,各个样品的特征及所属于的类别构成了聚类问题的解空间。编码是应用遗传算法时要解决的首要问题,也是关键问题。它决定了个体的染色体中基因的

排列次序,也决定了遗传空间到解空间的变換解码方法。编码的方法也影响到了遗传算子(选择、交叉、变异)的计算方法。好的编码方法能够大大提高遗传算法的效率。

如何确定编码有没有统一的标准,Dejong 曾给出两条参考规则。

① 有意义积木块编码原则:易于产生与所求问题相关,且具有低阶、短定义、长模式的编码方案。“模式”是指具有某些基因相似性的个体集合,而具有最短定义长度、低阶、确实硬度较高的模式称为构造优良个体的积木块或基因块。该原则理解为应采用易于生成高适应度个体的编码方案。

② 最小字符集编码原则:能使问题得到自然表示,或其描述具有最小编码字符集的编码方案。

这两条仅仅是指导规则,并不一定适应于所有场合。在实际应用中对编码方法、遗传算子等应统一考虑,以获得一种描述方便、计算效率高的方案。

常用的编码方法有以下几种。

(1) 二进制编码

二进制编码是遗传算法编码中最常用的方法。其编码符号集是二值符号集 $\{0,1\}$,其个体基因是二值符号串。

二进制编码中符号串的长度与问题的求解精度相关。设某一参数 x 的变化范围是 $[a, b]$,编码长度为 n ,则编码精度为 $\frac{b-a}{2^n-1}$ 。

二进制编码、解码操作简单易行,遗传操作便于实现,符合最小字符集编码原则。

(2) 符号编码方法

符号编码方法是指个体染色体串中的基因值取自一个无数值意义,只有代码含义的符号集。这个符号集可以是一个字母表,如 $\{A, B, C, \dots\}$,也可以是一个序号表 $\{1, 2, 3, \dots\}$,其优点是符合有意义的积木块原则,便于在遗传算法中利用所求问题的专业知识。

(3) 浮点数编码

此编码方案中个体的每个基因值是一个浮点数,一般采用决策变量的真实值。该方法适合在遗传算法中表示较大的数,应用于高精度的遗传算法,搜索空间较大,改善了算法的复杂性。

12.2.2 适应度函数

在遗传算法中,模拟自然选择的过程主要通过评估函数 $\text{CalObjValue}(\cdot)$ 和适应度函数 $\text{CalFitnessValue}(\cdot)$ 来实现。前者计算每个个体优劣的绝对值,后者计算每个个体相对于整个群体的相对适应性。个体适应度的大小决定了它继续繁衍还是消亡。适应度高的个体被复制到下一代的可能性高于适应度低的个体。

适应度函数是整个遗传算法中极为关键的一部分。好的适应度函数能够指导我们从非最优的个体进化到最优个体,并且能够用来解决一些遗传算法中的问题,如过早收敛与过慢结束的矛盾。

如果个体的适应度值很高,大大高于个体适应度均值,它将得到更多的机会被复制,所以有可能在没有达到最优解,甚至没有得到可接受解时,就因为某个或某些个体的副本充斥整个

群体而过早地收敛到局部最优解,失去了找到全局最优解的机会。这就是所谓的过早收敛问题。要解决过早收敛问题,就要调整适应度函数,对适应度值的范围进行压缩,防止那些“过于适应”的个体过早地在整个群体中占据统治地位。

与之相对应,遗传算法中还存在着结束缓慢的问题。也就是说,在迭代许多代以后,整个种群已经大部分收敛,但是还没有得到稳定的全局最优解。整个种群的平均适应度值较高,而且最优个体的适应度值与全体适应度均值间的差别不大,这就导致没有足够的力量推动种群遗传进化找到最优解。解决这个问题的方法是扩大适应度函数值的范围,拉大最优个体适应度值与群体适应度均值的距离。解决这个问题的方法还有适应度函数缩放方法、适应度函数排序法、适应度窗口技术、锦标赛选择等方法。

12.2.3 遗传算子

1. 选择算子

遗传算法中的“选择”算子就是用来确定如何从父代群体中按照某种方法,选择哪些个体作为子代的遗传算子。选择算子建立在对个体适应度进行评价的基础上,目的是避免基因损失,提高全局收敛性和计算效率。常用选择算子的操作方法有以下几种。

① 赌轮选择方法

赌轮选择方法又称比例选择方法,其基本思想是个体被选择的几率与其适应度值大小成正比。由于选择算子是随机操作的,这种算法的误差比较大,有时适应度最高的个体也不会被选中。各个样品按照其适应度值占总适应度值的比例组成面积为1的一个圆盘,指针转动停止后,指向的个体将被复制到下一代,适应度高的个体被选中的几率大,但是适应度低的个体也有机会被选中,这样有利于保持群体的多样性。

② 排序选择法

在计算每个个体的适应度之后,根据适应度大小对群体中的个体排序,再将事先设计好的概率表分配给各个个体,所有个体按适应度大小排序,选择概率与适应度无关,仅与序号相关。

③ 最优保存策略

最优保存策略的基本思想是:适应度最高的个体尽量保留到下一代群体中。其操作过程如下:

① 找出当前群体中适应度最高和最低的个体。

② 若当前个体适应度比总的迄今为止最好的个体适应度还要高,则用当前最优个体替代总的最优个体。

③ 用迄今为止最好个体替换最差个体。

最优保存策略保证最优个体不被破坏,能够被复制到下一代,是遗传算法收敛性的一个重要保证条件。另一方面,它也会使一个局部最优解不易被淘汰而迅速扩散,导致算法的全局搜索能力不强。

2. 交叉算子

在进化算法中,交叉是遗传算法所独有的方法,用于保留原始性特征。遗传算法交叉算子

模仿自然界有性繁殖的基因重组过程,其作用在于将原有的优良基因遗传给下一代个体,并生成包含更复杂结构的新个体,交叉操作一般分为以下几个步骤:

① 从交配池中随机地取出要交配的一对个体。

② 根据位串长度 L 对要交配的这对个体,随机地选取 $[1, L]$ 中的一个或多个整数作为交叉位。

③ 根据交叉概率 P_c ($0 < P_c \leq 1$) 实时交叉操作,配对个体在交叉位置相互交换各自的部分内容,从而形成一个新的个体。

通常使用的交叉算子有一点交叉、两点交叉、多点交叉和一致交叉等。

(1) 一点交叉

一点交叉是指在染色体中随机选择一个交叉点,如图 12-4(a) 所示,交叉点在第五位,然后把第一个父辈的交叉点前的位串和第二个父辈交叉点及其以后的位串,组成一个新的染色体,第二个父辈的交叉点前的位串和第一个父辈的交叉点及其以后的位串组合成另一个新的染色体,如图 12-4(b) 所示。

2	3	2	4	1	1	3	2	4	3	2	1	父代1
2	1	1	4	4	3	2	3	2	1	4	2	父代2

(a) 交叉点在父代染色体的第五位

2	3	2	4	4	3	2	3	2	1	4	2	子代1
2	1	1	4	1	1	3	2	4	3	2	1	子代2

(b) 交叉后得到的子代个体

图 12-4 一点交叉示意图

(2) 两点交叉

两点交叉是指在父代的染色体中随机选择两个交叉点,如图 12-5(a) 所示,然后交换父代染色体中交叉点间的基因,得到下一代个体,如图 12-5(b) 所示。

2	3	2	4	1	1	3	2	4	3	2	1	父代1
2	1	1	4	4	3	2	3	2	1	4	2	父代2

(a) 父辈的交叉点在第五位和第九位

2	3	2	4	4	3	2	3	4	3	2	1	父代1
2	1	1	4	1	1	3	2	2	1	4	2	父代2

(b) 交叉后得到的子代个体

图 12-5 两点交叉示意图

(3) 一致交叉

在一致交叉中,子代的每一位随机地从两个父代中的对应位取得。

3. 变异算子

变异操作模拟自然界生物体进化中,染色体上某位基因发生的突变现象,从而改变染色体的结构和物理性状。变异是遗传算法中保持物种多样性的一个重要途径。它以一定的概率选择个体染色体中的某一位或几位,随机地改变该位基因值,以达到变异的目的。

在遗传算法中,由于算法执行过程中的收敛现象,可能使整个种群的染色体上的某位或某几位都收敛到固定值。如果整个种群所有的染色体中有 n 位取值相同,那么单纯的交叉算子所能够达到的搜索空间只占整个搜索空间的 $(1/2)^n$,大大降低了搜索能力,所以,引进变异算子改变这种情况是必要的。生物学家一般认为变异是更为重要的进化方式,并且认为只通过选择与变异就能够进行生物进化的过程。

12.3 控制参数的选择

在遗传算法的运行过程中,存在着对其性能产生重大影响的一组参数,这组参数在初始阶段和群体进化过程中需要合理地选择和控制在,以使遗传算法以最佳的搜索轨迹达到最优解。主要参数有染色体位串长度 L ,群体规模 n ,交叉概率 P_c 和变异概率 P_m 。

(1) 位串长度 L

位串长度的选择取决于特定问题解的精度。要求精度越高,位串越长,但需要更多的计算时间。为了提高运行效率,可采用变长位串的编码方法。

(2) 群体规模 n

大群体含有较多的模式,为遗传算法提供了足够的模式采样容量,可以改善遗传算法的搜索质量,防止成熟前收敛。但是大群体增加了个体适应性的评价计算量,从而降低了收敛速度。一般情况下专家建议 $n = 20 \sim 200$ 。

(3) 交叉概率 P_c

交叉概率控制着交叉算子使用的频率,在每一代新的群体中,需要对 $P_c \cdot n$ 个个体的染色体结构进行交叉操作。交叉概率越高,群体中结构的引入就越快,已获得的优良基因结果的丢失速度也相应地提高了,而交叉概率太低则可能导致搜索阻滞。一般取 $P_c = 0.6 \sim 1.0$ 。

(4) 变异概率 P_m

变异操作是保持群体多样性的手段,交叉结束后,中间群体中的全部个体位串上的每位基因按变异概率 P_m 随机改变,因此每代中大约发生 $P_m \cdot n \cdot L$ 次变异。变异概率太小,可能使某些基因位过早地丢失信息而无法恢复;而变异(变异)概率过高,则遗传算法将变成随机搜索。一般取 $P_m = 0.005 \sim 0.05$ 。

实际上,上述参数与问题的类型有着直接的关系。问题的目标函数越复杂,参数选择就越困难。从理论上讲,不存在一组适应于所有问题的最佳参数值,随着问题特征的变化,有效参数的差异往往非常显著。如何设定遗传算法的控制参数,以使遗传算法的性能得到改善,还需要结合实际问题的深入研究。

12.4 基于遗传算法的聚类分析

1. 问题提出

许多学科要根据所测得的相似数据进行分类,把探测数据归入到各个聚合类中,且在同一个聚合类中的模式比不同聚合类中的模式更相似,从而对模式间的相互关系做出估计。聚类分析的结果可以被用来对数据提出初始假设,分类新数据,测试数据的同类型及压缩数据。聚类问题的特点是事先不了解一批样品中的每一个样品的类别或者其他的先验知识,而唯一的分类依据是样品的特性,利用样品的特性来构造分类器。聚类算法的重点是寻找特征的相似性。

一幅图像中含有多个物体,在图像中进行聚类分析需要对不同的物体分割标识,如图1-6所示,手写了(1,2,3,4,4,2,3,3,4,4,4)共12个待分类样品,要分成4类,如何让计算机自动将这12个样品归类呢?遗传算法在解决这种聚类问题上表现非常出色,它不仅运算速度快,而且准确率高。本节介绍用遗传算法解决聚类问题的实现方法。

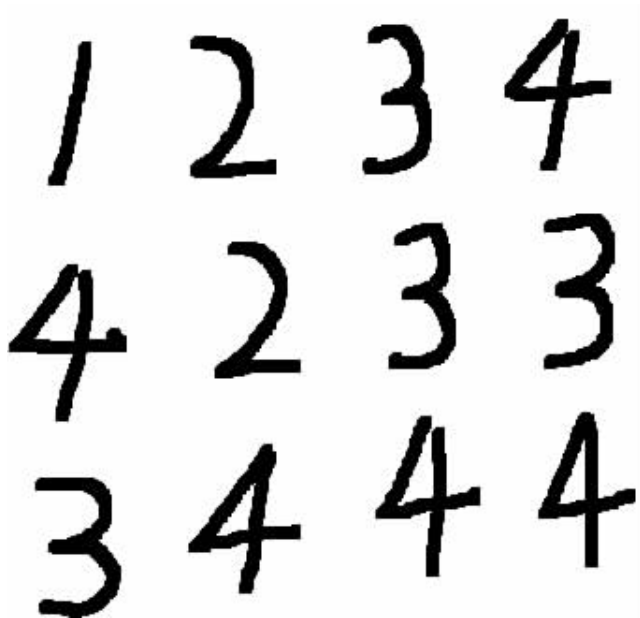


图 12-6 待聚的样品数字

2. 染色体的构造

对图 12-6 所示的 12 个样品进行编号,如图 12-7 所示,在每个样品的右上角,不同的样品编号不同,而且编号始终固定。

采用符号编码,位串长度 L 取 12 位,基因代表样品所属的类号(1~4),基因位的序号代表样品的编号,基因位的序号是固定的,也就是说某个样品在染色体中的位置是固定的,而每个样品所属的类别随时在变化。如果基因位为 n ,则其对应第 n 个样品,而第 n 个基因位所指向的基因值代表第 n 个样品的归属类号。

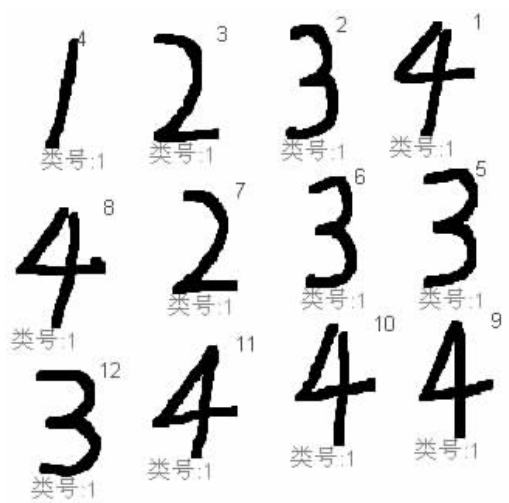


图 12-7 待测样品的编号

每个个体包含一种分类方案。设初始某个个体的染色体编码为:(2,3,2,1,4,4,2,3,1,3,2,3),其含义为:第1、3、7、11个样品被分到第2类;第2、8、10和12个样品被分到第3类;第4、9个样品被分到第1类;第5、6个样品属于第4类,这时还处于假设分类情况,不是最优解,如表12-1所示。

表 12-1 初始某个染色体编码

样品值	(4)	(3)	(2)	(1)	(3)	(3)	(2)	(4)	(4)	(4)	(4)	(3)
基因值 (分类号)	2	3	2	1	4	4	2	3	1	3	2	3
基因位	1	2	3	4	5	6	7	8	9	10	11	12
样品编号	1	2	3	4	5	6	7	8	9	10	11	12

经过遗传算法找到的最优解,如图12-8所示。遗传算法找到的最优染色体编码如表12-2所示。通过样品值与基因值比较对照,就会发现相同的数据被归为一类,分到相同的类号,而且全部正确。

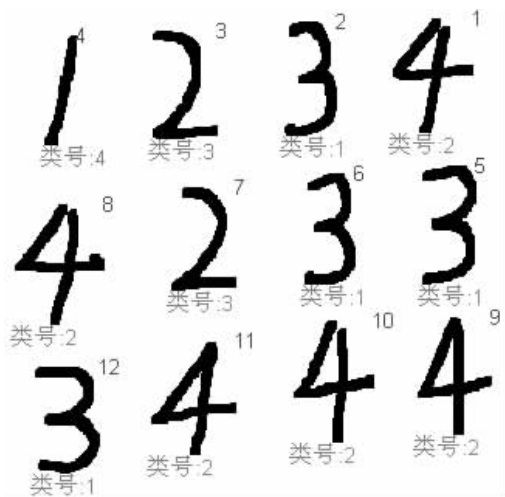


图 12-8 遗传算法找到的最优解

表 12-2 遗传算法最终找到的最优染色体编码

样品值	(4)	(3)	(2)	(1)	(3)	(3)	(2)	(4)	(4)	(4)	(4)	(3)
基因值 (分类号)	2	1	3	4	1	1	3	2	2	2	2	1
基因位	1	2	3	4	5	6	7	8	9	10	11	12
样品编号	1	2	3	4	5	6	7	8	9	10	11	12

3. 设定评估函数

评估函数 CalObjValue() 结果为评估值, 代表每个个体优劣的程度。

对初始群体中的每个染色体计算它们的评估值 $m_pop(i).value$, 实现步骤如下:

① 通过人工干预获得聚类类别总数, $centerNum$ 为聚类类别总数 ($2 \leq centerNum \leq N$, N 是总的样品个数)。

② 找出染色体中相同类号的样品, $X^{(\omega_i)}$ 表示属于第 i 个类的样品。

③ 统计每一类的样品个数, N_i 是第 i 个类别的个数, 样品总数为: $N = \sum_{i=1}^{centerNum} N_i$ 。

④ 计算每一类的中心 $\overline{X^{(\omega_i)}}$, $\overline{X^{(\omega_i)}}$ 是第 i 个类中心,

$$\overline{X^{(\omega_i)}} = \frac{1}{N_i} \sum_{k=1}^{N_i} X_k^{(\omega_i)}, (i = 1, 2, \dots, centerNum);$$

⑤ 同一个类内计算每一个样品到中心的距离, 并将它们累加求和。采用 K-均值模型为聚类模型, K-均值的理论已经在第 9 章介绍过。计算公式如下:

$$D_i = \sum_{j=1}^{N_i} \|X_j^{(\omega_i)} - \overline{X^{(\omega_i)}}\|^2 \quad (12-1)$$

显然当聚类类别总数的数目 $centerNum = N$ 时, 累加和 $\sum D_i$ 为 0。因此, 当聚类数目 $centerNum$ 不定时, 我们必须对目标函数进行修正。实际上, 式(11-1)仅为类内距离之和, 因此, 可以使用类内距离与类间距离之和作为目标函数, 即

$$D = \min \left[w \cdot \sum_{i=1}^{centerNum} \sum_{j=1}^{N_i} \|X_j^{(\omega_i)} - \overline{X^{(\omega_i)}}\|^2 + \sum_{i=1}^{centerNum} \sum_{j=i}^{centerNum} \|\overline{X^{(\omega_i)}} - \overline{X^{(\omega_j)}}\|^2 \right] \quad (12-2)$$

其中, w 是权重, 反映决策者的偏爱; 当 w 变大时, 聚类总数数目将变大, 反之聚类总数数目将变小。

⑥ 将不同类计算出的 D_i 求和赋给 $m_pop(i).value$, 以 $m_pop(i).value$ 作为评估值。

$$m_pop(i).value = \sum_{i=1}^{centerNum} \sum_{j=1}^{N_i} \|X_j^{(\omega_i)} - \overline{X^{(\omega_i)}}\|^2 = \sum_{i=1}^{centerNum} D_i \quad (12-3)$$

$m_pop(i).value$ 越小, 说明这种分类方法的误差越小, 该个体被选择到下一代的几率也就越大。

4. 设定适应度函数

适应度函数是整个遗传算法中极为关键的一部分。好的适应度函数能够从非最优的个体进化到最优个体, 能够解决过早收敛与过慢结束的矛盾。

适应度函数 CalFitnessValue() 结果代表每个个体相对于整个群体的相对适应性。个体适应度的大小决定了它继续繁衍还是消亡。适应度高的个体被复制到下一代的可能性高于适应度低的个体。

(1) 选择机制在遗传算法中存在的问题

以 $m_pop(i).value$ 作为适应度,其选择机制在遗传算法中存在两个问题:

① 群体中极少数适应度相当高的个体被迅速选择、复制遗传,引起算法提前收敛于局部最优解。

② 群体中个体适应度彼此非常接近,算法趋向于纯粹的随机选择,使优化过程趋于停止。

这里不以 $m_pop(i).value$ 直接作为该分类方法的适应度值,采用的方法是适应度排序法。不管个体的 $m_pop(i).value$ 是多少,被选择的几率只与序号有关。这样避免了一代群体中过于适应或过于不适应个体的干扰。

(2) 适应度函数计算步骤

① 按照原始的 $m_pop(i).value$ 由小到大排序,依次编号为 $1, 2, \dots, popSize$ 。index 是排序序号。

② 计算适应度值:

$$m_pop(i).fitness = a(1 - a)^{index - 1} \quad (12-4)$$

其中, a 取值范围是 $(0, 1)$, 取 $a = 0.6$ 。

5. 遗传算子

(1) 选择算子

建立选择数组 $cFitness(popSize)$, 循环统计从第 1 个个体到第 i 个个体适应度值之和占所有个体适应度值总和的比例 $cFitness(i)$, 以 $cFitness(i)$ 作为选择依据。

$$cFitness(i) = \frac{\sum_{k=1}^i m_pop(k).fitness}{S} \quad (12-5)$$

式中, $S = \sum_{i=1}^{popSize} m_pop(i).fitness$

循环产生随机数 p , 当 $p < cFitness(i)$ 时, 对应的第 i 个个体复制到下一代中, 直到生成中间群体。例如, 由 4 个个体组成的群体, $a = 0.6$, $cFitness(i)$ 计算方式如表 12-3 所示。

表 12-3 选择依据计算方式

index	适应度值: $a(1 - a)^{index - 1}$	选择依据: $cFitness(i)$
1	$0.6(1 - 0.6)^0 = 0.6$	$0.6/S$
2	$0.6(1 - 0.6)^1 = 0.6 \cdot 0.4 = 0.24$	$(0.6 + 0.6 \cdot 0.4)/S$
3	$0.6(1 - 0.6)^2 = 0.6 \times 0.4 \times 0.4 = 0.096$	$(0.6 + 0.6 \cdot 0.4 + 0.6 \times 0.4 \times 0.4)/S$
4	$0.6(1 - 0.6)^3 = 0.6 \times 0.4 \times 0.4 \times 0.4 = 0.0384$	$(0.6 + 0.6 \cdot 0.4 + 0.6 \times 0.4 \times 0.4 + 0.6 \times 0.4 \times 0.4 \times 0.4)/S$

(2) 交叉算子

以概率 P_c 生成一个“一点交叉”的交叉位 point, 随机不重复地从中间群体中选择两个个体, 对交叉位后的基因进行交叉运算, 直到中间群体中所有个体都被选择到。

(3) 变异算子

对所有个体, 循环每一个基因位, 产生随机数 p , 当概率 $p < P_m$ 时, 对该位基因进行变异运

算,随机产生 $1 \sim \text{centerNum}$ 之间的一个数赋值给该位,生成子代群体。

变异概率一般很小, P_m 在 $0.001 \sim 0.1$ 之间,如果变异概率过大,会破坏许多优良品种,也可能无法得到最优解。

6. 实现步骤

① 设置相关参数。

初始化初始种群总数 $\text{popSize} = 200$,交叉概率 0.6 ,变异概率 0.05 。从对话框得到用户输入的最大迭代次数 MaxGeneration ,聚类中心数目 centerNum 。

② 获得所有样品个数及特征。

③ 群体初始化。

④ 计算每一个个体的评估值 $m_pop(i).value$ 。

⑤ 计算每一个个体的适应度值 $m_pop(i).fitness$ 。

⑥ 生成下一代群体。

➤ 选择算子:建立适应度数组 $cFitness(\text{popSize})$,计算 $cFitness(i)$ 。循环产生随机数 p ,当 $p < cFitness(i)$ 时,对应的个体复制到下一代中,直到生成 200 个中间群体。

➤ 交叉算子:以概率 $P_c = 0.6$ 生成一个“一点交叉”的交叉位 point ,随机从中间群体中选择两个个体,对交叉位后的基因进行交叉运算,直到中间群体中所有个体都被选择过。

➤ 变异算子:对所有个体,循环每一个基因位,产生随机数 p ,当概率 $p < P_m = 0.05$ 时,对该位基因进行变异运算,随机产生 $1 \sim \text{centerNum}$ 之间的一个数赋值给该位,生成子代群体。

⑦ 再次调用 $\text{EvaPop}()$ 函数对新生成的子代群体(200 个)进行评估。

⑧ 调用 $\text{FindBW}()$ 函数保留精英个体,若新生成的子代群体中的最优个体 D 值低于总的最优个体的 D 值(相互之间距离越近, D 越小),则用当前最好的个体替换总的最好的个体,否则用总的最好个体替换当前最差个体。

⑨ 若已经达到最大迭代次数,则退出循环,否则到第⑥步“生成下一代群体”继续运行。

⑩ 将总的最优个体的染色体解码,返回给各个样品的类别号。

遗传算法应用于聚类分析的程序总体流程图如图 12-9 所示。

7. 编程代码

在本书提供的基于遗传算法聚类具体功能在 GA. m 中定义:

(1) 初始化各个参数

```
%%%%%%%%%%
% 函数名称:C_GA()
% 参数:m_pattern:样品特征库;patternNum:样品数目
% 返回值:m_pattern:样品特征库
% 函数功能    按照遗传算法对全体样品进行聚类
%%%%%%%%%
function [ m_pattern ] = C_GA( m_pattern,patternNum )
[ centerNum MaxGeneration ] = InputClassDlg();% 获得类中心数和最大迭代次数
disType = DisSelDlg();% 获得距离计算类型
popSize = 200;% 种群大小
```

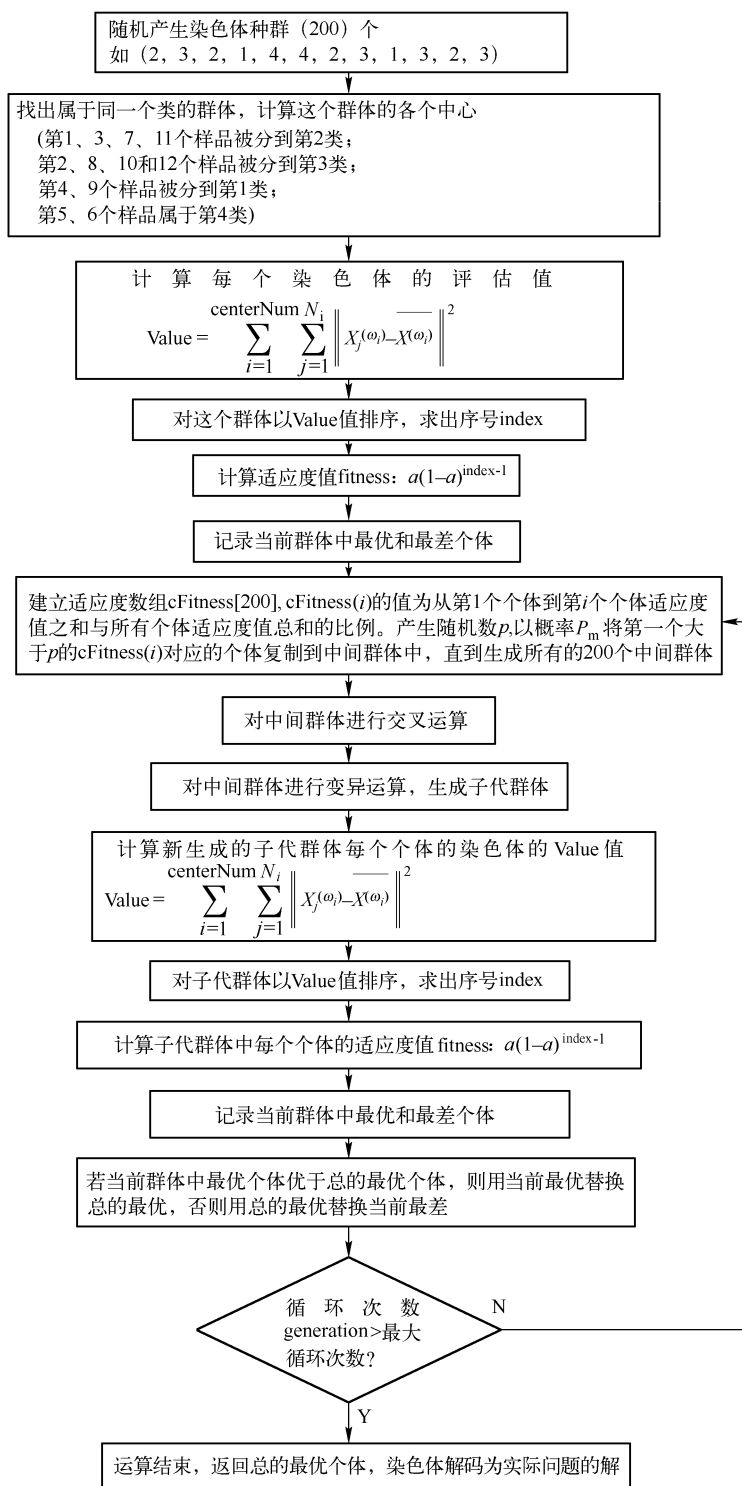


图 12-9 基于遗传算法的聚类问题流程图

```
% 初始化种群体结构
for i = 1 : popSize
    m_pop(i). string = ceil( centerNum. * rand(1, patternNum) ); % 初始化个体位串
    m_pop(i). index = -1; % 索引
    m_pop(i). value = 0; % 评估值
    m_pop(i). fitness = 0; % 适应度
end

% 初始化全局最优最差个体
cBest = m_pop(1); % 其中 cBest 的 index 属性记录最优个体出现在第几代中
cWorst = m_pop(1);
pc = 0.6; % 交叉概率
pm = 0.05; % 变异概率
```

(2) 评价群体

```

[m_pop] = CalObjValue(m_pop, popSize, patternNum, centerNum, m_pattern, disType); % 计算个
体的评估值

[m_pop] = CalFitnessValue(m_pop, popSize); % 计算个体的适应度

[cBest, cWorst] = FindBW(m_pop, popSize, cBest, cWorst, generation); % 寻找最优个体,更新
总的最优个体 Generation 当前代。

% 函数名称: CalObjValue()

% 参数: m_pop: 种群结构; popSize: 种群规模; patternNum: 样品数目;

%       enterNum: 类中心数; m_pattern: 样品特征库;

%       disType: 距离类型

% 返回值: m_pop: 种群结构;

% 函数功能: 计算个体的评估值

function [m_pop] = CalObjValue(m_pop, popSize, patternNum, centerNum, m_pattern, disType)

    global Nwidth;

    for i = 1:popSize

        for j = 1:centerNum % 初始化聚类中心

            m_center(j).index = i;

            m_center(j).feature = zeros(Nwidth, Nwidth);

            m_center(j).patternNum = 0;

        end

        % 计算聚类中心

        for j = 1:patternNum

            m_center(m_pop(i).string(1, j)).feature = m_center(m_pop(i).string(1, j)).feature +

            m_pattern(j).feature;

            m_center(m_pop(i).string(1, j)).patternNum =

            m_center(m_pop(i).string(1, j)).patternNum + 1;

        end
    end
end

```

```

d = 0;
for j = 1:centerNum
    if(m_center(j). patternNum ~ = 0)
        m_center(j). feature = m_center(j). feature/m_center(j). patternNum;
    else
        d = d + 1;
    end
end
m_pop(i). value = 0;
% 计算个体评估值
for j = 1:patternNum

    m_pop(i). value = m_pop(i). value + GetDistance( m_center( m_pop(i). string(1,j) ),
        m_pattern(j) ,disType)^2;
end
m_pop(i). value = m_pop(i). value + d;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 函数名称: CalFitnessValue( )
% 参数: m_pop: 种群结构; popSize: 种群规模;
% 返回值: m_pop: 种群结构;
% 函数功能: 计算个体的适应度
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ m_pop ] = CalFitnessValue( m_pop, popSize)
    for i = 1:popSize
        m_pop(i). index = - 1;
    end
    % 按照 value 大小排序
    for i = 1:popSize
        index = 1;
        for j = 1:popSize
            if( m_pop(j). value < m_pop(i). value && i ~ = j)
                index = index + 1;
            elseif( m_pop(i). value == m_pop(j). value && m_pop(j). index ~ = - 1 && i ~ = j)
                index = index + 1;
            end
        end
        m_pop(i). index = index;
    end
    a = 0.6;
    for i = 1:popSize
        m_pop(i). fitness = a * (1 - a)^( m_pop(i). index - 1 );
    end
end

```

```

end

%% 函数名称:FindBW()
%% 参数:m_pop:种群结构;popSize:种群规模;
%      cBest:最优个体;cWorst:最差个体;generation:当前代数
% 返回值:cBest:最优个体;cWorst:最差个体;
% 函数功能:寻找最优个体,更新总的最优个体
function [ cBest,cWorst ] = FindBW( m_pop,popSize,cBest,cWorst,generation)
% 初始化局部最优个体
best = m_pop(1);
worst = m_pop(1);
for i = 2:popSize
    if( m_pop(i).value < best.value)
        best = m_pop(i);
    elseif( m_pop(i).value > worst.value)
        worst = m_pop(i);
    end
end
if( generation == 1 )
    cBest = best;
    cBest.index = 1;
else
    if( best.value < cBest.value)
        cBest = best;
        cBest.index = generation;
    end
end
end

```

(3) 生成下一代群体

```

[m_pop] = Selection( m_pop,popSize)//选择算子

[m_pop] = Crossover( m_pop,popSize,pc,patternNum)//交叉算子

[m_pop] = Mutation( m_pop,popSize,pm,patternNum,centerNum)//变异算子

```

① 选择算子

建立适应度数组 $cFitness[popSize]$, $cFitness(i)$ 的值为从第 1 个个体到第 i 个个体适应度值之和占所有个体适应度值总和的比例。即:设 $S = \sum_{i=1}^{popSize} m_pop(i).value$, 则

$$cFitness(i) = \frac{\sum_{k=1}^i m_pop(k).value}{S},$$

产生随机数 p , 以概率 P_m 将第一个大于 p 的 $cFitness(i)$ 对应的个体复制到下一代中, 直

到生成 200 个中间群体。编程代码如下。

```
% 函数名称: Selection()
% 参数: m_pop: 种群结构; popSize: 种群规模;
% 返回值: m_pop: 种群结构
% 函数功能: 选择操作

function [ m_pop ] = Selection( m_pop, popSize )
    cFitness = zeros( 1, popSize );
    for i = 1 : popSize
        if ( i == 1 )
            cFitness( i ) = m_pop( i ). fitness;
        else
            cFitness( i ) = cFitness( i - 1 ) + m_pop( i ). fitness;
        end
    end
    cFitness = cFitness / cFitness( popSize );
    for i = 1 : popSize
        p = rand;
        index = 1;
        while( cFitness( index ) < p )
            index = index + 1;
        end
        newPop( i ) = m_pop( index );
    end
    m_pop = newPop;
```

② 交叉算子

以概率 P_c 生成一个“一点交叉”的交叉位 point, 随机的从中间群体中选择两个个体, 对交叉位后的基因进行交叉运算, 直到中间群体中所有个体都被选择过。

编程代码:

```
% 函数名称: Crossover( )
% 参数: m_pop: 种群结构; popSize: 种群规模; pc: 交叉概率;
%      patternNum: 样品数量
% 返回值: m_pop: 种群结构
% 函数功能: 交叉操作

function [ m_pop ] = Crossover( m_pop, popSize, pc, patternNum )
    % 交叉操作
    for i = 1 : popSize / 2
        p = rand;
```

```

        if(p < pc)
            point = fix( rand * patternNum); % 生成随机位
            for j = point + 1 : patternNum % 交叉
                temp = m_pop(2 * i - 1). string(1,j);
                m_pop(2 * i - 1). string(1,j) = m_pop(2 * i). string(1,j);
                m_pop(2 * i). string(1,j) = temp;
            end
        end
    end
end
end

```

③ 变异算子

对所有个体, 以概率 P_m 对染色体中的每一位进行变异运算, 使该位随机的生成 1-center-num 之间的一个数, 生成子代群体。

```

% 函数名称: Mutation( )
% 参数: m_pop: 种群结构; popSize: 种群规模; pm 变异概率;
%       patternNum: 样品数量; centerNum: 类中心数;
% 返回值: m_pop: 种群结构
% 函数功能: 变异操作
function [ m_pop ] = Mutation( m_pop, popSize, pm, patternNum, centerNum)
    for i = 1 : popSize
        for j = 1 : patternNum
            p = rand;
            if(p < pm)
                m_pop(i). string(1,j) = fix( rand * centerNum + 1 );
            end
        end
    end
end
end

```

(4) 评估子代群体

再次对新生成的子代群体进行评估。

(5) 保留精英个体

若新生成的子代群体中的最优个体适应度值高于总的最优个体的适应度值, 则用当前最好的个体替换总的最好的个体。

(6) 判断循环条件

若已经达到最大循环次数, 则退出循环, 否则到第(3)步“生成下一代群体”继续运行。

(7) 获得最优解

将总的最优个体的染色体解码, 返回给各个样品的类别号。

```

for i = 1 : patternNum

```



```

m_pattern(i).category = cBest.string(1,i);
end

```

8. 效果图

这里给读者提供两个实例效果图,一个是基于数字聚类结果图,如图 12-10 所示,另一个是基于图形聚类结果,如图 12-11 所示。这两个图也比较复杂,但从结果可以看出,应用遗传算法解决聚类问题效果非常好。注意:图右上角显示样品编号,左下角显示该样品所属类别。



图 12-10 遗传算法应用于数字聚类分析

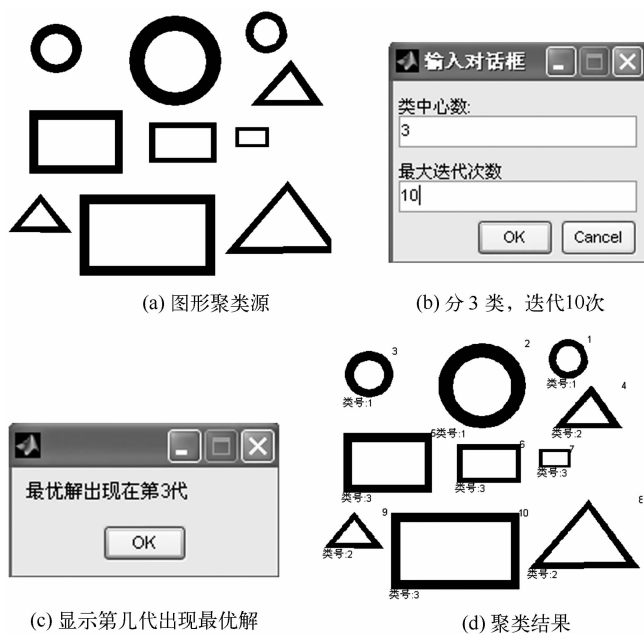


图 12-11 遗传算法应用于图形聚类分析

本章小结

本章介绍了遗传算法基本概念,包括遗传算法的构成要素、染色体的编码等内容;介绍了遗传算子的实现方法,包括选择算子、交叉算子和变异算子以及评估函数、适应度函数的设定方法,还介绍了遗传算法进行问题求解过程以及控制参数的选择,并着重介绍了遗传算法在聚类问题中的实现方法和步骤。

习题 12

1. 简述遗传算法的构成要素。
2. 简述选择算子、交叉算子和变异算子的作用。
3. 简述遗传算法进行问题求解步骤。
4. 如果一幅位图当中包含有 15 个样品,分成 5 类,叙述染色体的编码方法,给出一种可能的染色体编码方案,染色体长度应该设为多长? 如何设定评估函数? 如何设定适应度函数?
5. 叙述遗传算法在聚类问题中的实现方法和步骤。

第 13 章 蚁群算法聚类分析

本章要点:

- ☑ 蚁群算法的基本概念
- ☑ 聚类数目未知的蚁群聚类算法
- ☑ 聚类数目已知的蚁群聚类算法

13.1 蚁群算法的基本原理

蚁群算法(Ant Colony Optimization, ACO)是近年来才提出的一种基于种群寻优的启发式搜索算法,由意大利学者 M. Dorigo 等于 1991 年首先提出。该算法受到自然界中真实蚁群集体行为的启发,利用真实蚁群通过个体间的信息传递、搜索从蚁穴到食物间的最短路径的集体寻优特征,来解决一些离散系统中优化的困难问题。目前,该算法已经被用于求解旅行商问题(Traveling Salesman Problem, TSP)、指派问题以及调度问题等,取得了较好的实验结果。

1. 蚁群算法的基本思想

蚁群算法是受真实的蚂蚁的群体合作行为而提出的一种随机搜索算法,与其他进化算法相似,通过对候选解组成的群体进化来寻求最优解。各候选解根据积累的信息不断调整自身结构并且通过信息与其他候选解进行交流,以产生更好的解。

作为一种随机优化方法,蚁群算法不需要任何先验知识,最初只是随机地选择搜索路径,随着对解空间的了解,搜索更加具有规律性,并逐渐得到全局最优解。蚁群算法的搜索机制主要包括以下方面:

(1) 存在一种群体中蚂蚁相互交流通信的机制,通常表示为信息素

蚂蚁能在其走过的路径上分泌一种化学物质 Pheromone(信息素),通过这种方式形成信息素轨迹。蚂蚁在运动过程中能够感知这种物质的存在及其强度,并以此指导自己的运动方向,使蚂蚁倾向于朝着该物质强度高的方向移动。信息素轨迹可以使蚂蚁找到它们返回食物源(或蚁穴)的路径。当同伴蚂蚁进行路径选择时,会根据路径上的信息素进行选择,这样信息素成为蚂蚁之间通信的媒介。

(2) 蚂蚁的集群活动

每只蚂蚁在寻找食物源的时候只贡献了非常小的一部分,但整个蚁群的行为却表现出了具有找出最短路径的能力。由大量蚂蚁组成的蚁群的集体行为表现出一种信息的正反馈现象:某一路径上走过的蚂蚁越多,导致信息素强度增大,该路径对后来的蚂蚁就越有吸引力,即一只蚂蚁选择一条路径的概率随着以前选择该路径的蚂蚁数量的增加而增大。而某些路径上通过的蚂蚁较少时,路径上的信息素就会随时间推移而逐渐蒸发。蚂蚁这种搜索路径的过程被称之为自催化行为或正反馈机制。因此,通过模拟这种机制即可使蚁群算法的搜索向最优

解推进。

(3) 利用当前信息进行路径选择的概率选择策略

蚁群算法中蚂蚁从一个节点移动到下一个节点的求解方法是利用概率选择策略实现的, 概率选择策略只利用当前的信息去预测未来的情况, 而不能利用未来的信息, 因此, 该选择策略利用的都是当前信息。

2. 蚁群算法的基本模型

由于基本的蚁群算法与旅行商问题的求解具有紧密的联系, 本节以求解旅行商问题为例说明蚁群算法的基本模型。

旅行商问题属于一种典型的组合优化问题, 定义是指给定 n 个城市集合以及两两城市之间的距离, 寻找一条具有最短长度的闭合路径, 该路径经过全部城市且每个城市经过一次。旅行商问题可由图论描述, 即给定图 $G = (V, A)$, 其中 V 为城市集合, A 为城市之间的支路集合, 已知各城市间的连接距离, 要求确定一条长度最短的 Hamilton 回路, 即当且仅当一次遍历所有城市的最短回路。下面介绍其基本算法。

(1) 蚁群算法的常见符号

为了更清楚地描述蚁群算法, 首先引入下列符号:

- ① $q_i(t)$: t 时刻位于城市 i 的蚂蚁个数;
- ② m : 蚁群中的全部蚂蚁个数, $m = \sum_{i=1}^n q_i(t)$;
- ③ τ_{ij} : 边 (i, j) 上的信息素强度;
- ④ η_{ij} : 边 (i, j) 上的能见度;
- ⑤ d_{ij} : 城市 i 与城市 j 之间的距离;
- ⑥ P_{ij}^k : 蚂蚁 k 由城市 i 向城市 j 转移的概率。

(2) 每只蚂蚁具有的特征

① 蚂蚁根据城市距离和连接边上信息素数量为变量的概率函数选择下一个将要访问的城市。

② 规定蚂蚁在完成一次循环以前, 不允许转到已访问的城市。

③ 蚂蚁在完成一次循环时, 在每一条访问的边 (i, j) 上释放信息素。

(3) 蚁群算法流程

基本蚁群算法流程如图 13-1 所示。

① 初始化蚁群: 初始化蚁群参数, 设置蚂蚁数量, 将蚂蚁置于 n 个顶点上, 初始化路径信息素。

② 蚂蚁移动: 蚂蚁依据前面蚂蚁所留下的信息素和自己的判断选择路径, 完成一次循环。

③ 释放信息素: 对蚂蚁所经过的路径按一定的

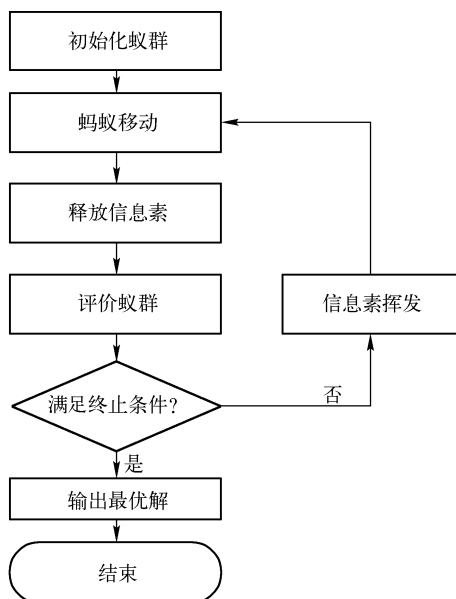


图 13-1 基本蚁群算法流程

比例释放信息素。

④ 评价蚁群:根据目标函数对每只蚂蚁的适应度做出评价。

⑤ 若满足终止条件,即最短路径,输出最优解。否则,算法继续进行。

⑥ 信息素的挥发:信息素会随着时间延续不断消散。

初始时刻,各条路径上的信息素相等,设 $\tau_{ij}(0) = C$ (C 为常数)。蚂蚁 k ($k=1,2,\dots,m$) 在运动过程中根据各条路径上的信息素决定转移方向。在 t 时刻,蚂蚁 k 在城市 i 选择城市 j 的转移概率 $P_{ij}^k(t)$ 为

$$P_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}^\alpha(t) \eta_{ij}^\beta(t)}{\sum_{s \in \text{allowed}_k} \tau_{is}^\alpha(t) \eta_{is}^\beta(t)}, & \text{若 } j \in \text{allowed}_k \\ 0, & \text{否则} \end{cases} \quad (13-1)$$

式中, $\text{allowed}_k = \{0, 1, \dots, n-1\} - \text{tabu}_k$ 表示蚂蚁 k 下一步允许选择的的城市, tabu_k 记录蚂蚁 k 当前所走过的城市, tabu_k 随进化过程做动态调整。由式(13-1)可知,转移概率 P_{ij}^k 与 $\tau_{ij}^\alpha \eta_{ij}^\beta$ 成正比。 η_{ij} 为能见度因数,用某种启发式算法得到,一般取 $\eta_{ij} = 1/d_{ij}$, α 和 β 为两个参数,反映了蚂蚁在运动过程中信息素轨迹和能见度在蚂蚁选择路径中的相对重要性。经过 n 个时刻,蚂蚁完成一次循环,各路径上信息素量根据下式调整

$$\tau_{ij}(t+n) = (1-\rho) \tau_{ij}(t) + \Delta \tau_{ij} \quad (13-2)$$

$$\Delta \tau_{ij} = \sum_{k=1}^m \Delta \tau_{ij}^k \quad (13-3)$$

式中, $\Delta \tau_{ij}^k$ 表示第 k 只蚂蚁在本次循环中留在路径 (i,j) 上的信息素量,其值视蚂蚁表现的优劣程度而定。路径越短,信息素释放的就越多; $\Delta \tau_{ij}$ 表示本次循环中路径 (i,j) 的信息素量的增量; ρ 为信息素轨迹的衰减系数,通常设置系数 $\rho < 1$ 来避免路径上信息素量的无限累加。

根据具体算法的不同, $\Delta \tau_{ij}$ 、 $\Delta \tau_{ij}^k$ 及 $P_{ij}^k(t)$ 的表达形式可以不同,要根据具体问题而定。M. Dorigo 曾给出三种不同模型,分别称为蚁周系统(ant-cycle system)、蚁量系统(ant-quantity system)、蚁密系统(ant-density system)。它们的差别在于求解 $\Delta \tau_{ij}^k$ 的方法不同。

在蚁密系统模型中

$$\Delta \tau_{ij}^k = \begin{cases} Q, & \text{若第 } k \text{ 只蚂蚁在本次循环中经过路径 } (i,j) \\ 0, & \text{否则} \end{cases} \quad (13-4)$$

式中, Q 为常量。

在蚁量系统模型中

$$\Delta \tau_{ij}^k = \begin{cases} Q/d_{ij}, & \text{若第 } k \text{ 只蚂蚁在本次循环中经过路径 } (i,j) \\ 0, & \text{否则} \end{cases} \quad (13-5)$$

在蚁周系统模型中

$$\Delta \tau_{ij}^k = \begin{cases} Q/L_k, & \text{若第 } k \text{ 只蚂蚁在本次循环中经过路径 } (i,j) \\ 0, & \text{否则} \end{cases} \quad (13-6)$$

式中, L_k 为第 k 只蚂蚁在本次循环中所走的路径长度。

在蚁密系统和蚁量系统中,蚂蚁在建立方案的同时释放信息素,利用的是局部信息;而蚁周系统是在蚂蚁已经建立了完整的轨迹后再释放信息素,利用的是整体信息。

3. 蚁群算法的特点

(1) 优点

① 蚁群算法本质上是一种模拟进化算法,结合了分布式计算、正反馈机制和贪婪式搜索算法,在搜索过程中不容易陷入局部最优,即在所定义的适应函数是不连续、非规则或有噪声的情况下,也能以较大的概率发现最优解,同时贪婪式搜索有利于快速找出可行解,缩短了搜索时间。

② 蚁群算法采用自然进化机制来表现复杂的现象,通过信息素合作而不是个体之间的通信机制,使算法具有较好的可扩充性,能够快速可靠地解决困难的问题。

③ 蚁群算法具有很高的并行性,非常适合于巨量并行机。

(2) 缺陷

① 通常该算法需要较长的搜索时间。由于蚁群中个体的运动是随机性的,当群体规模较大时,要找出一条较好的路径就需要较长的搜索时间。

② 蚁群算法在搜索过程中容易出现停滞现象,表现为搜索到一定阶段后,所有解趋向一致,无法对解空间进行进一步搜索,不利于发现更好的解。

因此,在针对不同的优化问题求解时,就需要设计不同的蚁群算法,合理地选择目标函数,信息更新和群体协调机制,尽量避免算法缺陷。

13.2 聚类数目已知的蚁群聚类算法

聚类分析是一种传统的多变量统计分类方法,用以探讨如何将所搜集的物体分类,使得相同群体内具有高度的相似性以及不同群体间具有高度的相异性。聚类分析的用途甚广,在科学数据探测、图像处理、模式识别、文档检索、医疗诊断、Web 分析、计算物理学等领域起着非常重要的作用。

聚类问题的本质是一个非线性规划问题,目前没有有效的算法解决这些问题。蚁群算法作为一种分布式寻优算法,已经展示了其优良的搜索最优解的能力,并具有其他通用型算法不具备的特性。由于蚁群算法能够应用于各种优化组合问题,因此可以用来解决聚类分析问题。基于蚁群算法的聚类算法大致分为“聚类数目已知”和“聚类数目未知”两类问题,本节和下一节将分别介绍这两类算法。

1. 问题提出

一幅图像中含有多个物体,在图像中进行聚类分析需要对不同的物体分割标识,如图 13-2 所示,手写了 12 个待分类样品,要分成 4 类,如何让计算机自动将这 12 个物体归类呢? 本节介绍用蚁群算法解决聚类问题的实现方法。

2. 蚂蚁的结构

在已知聚类数目的蚁群聚类算法中,每只蚂蚁都表示为一种可能的聚类结果。首先生成具有 m 只蚂蚁的蚁群,每只蚂蚁在搜索开始之前分配一个空的长度为样品个数 N 的解集 S ,解集中的第 i 个位置对应于第 i 个样品

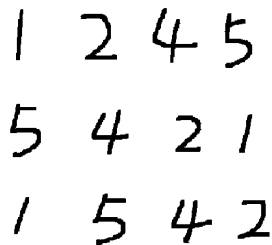


图 13-2 待聚类的样品数字

所属的类号。在搜索结束后,解集中的值表示的是第 i 个样品所归属的类。

针对图 13-2,分为 4 类的 12 个样品,设计蚂蚁的解集,假设蚂蚁 S_i 进行搜索后找到的解集,如表 13-1 所示。

表 13-1 某只蚂蚁的解集

样品号	1	2	3	4	5	6	7	8	9	10	11	12
蚂蚁 S_i	2	4	1	3	2	3	1	4	4	1	3	1

某一只蚂蚁 S_i 中的值表示的是第 1 个样品分到第 2 类,第 2 个样品分到第 4 类,……这是蚂蚁利用信息素把每个样品分到相应的类中后的解集。

3. 构造信息素矩阵

在将 12 样品分到 4 个类规模的聚类问题中,信息素是一个在迭代过程中不断更新的 12×4 的矩阵。在初始阶段,信息素值 τ 被初始化为同一个数值 τ_0 , τ_{ij} 代表样品 i 分配到它所属的类 j 的信息素值。例如,如表 13-2 所示的是一个将 12 样品分到 4 个类的初始信息素矩阵。

表 13-2 信息素矩阵

类别 j 样品 i	1	2	3	4
1	0.01	0.01	0.01	0.01
2	0.01	0.01	0.01	0.01
3	0.01	0.01	0.01	0.01
4	0.01	0.01	0.01	0.01
5	0.01	0.01	0.01	0.01
6	0.01	0.01	0.01	0.01
7	0.01	0.01	0.01	0.01
8	0.01	0.01	0.01	0.01
9	0.01	0.01	0.01	0.01
10	0.01	0.01	0.01	0.01
11	0.01	0.01	0.01	0.01
12	0.01	0.01	0.01	0.01

4. 构造目标函数

已知模式样品集 $\{X\}$ 有 N 个样品和 M 个模式分类 $\{S_j, j=1, 2, \dots, M\}$, 每个样品有 n 个特征。以每个模式样品到聚类中心的距离之和达到最小作为目标函数,其数学模型表示为

$$\min J(w, c) = \sum_{j=1}^m \sum_{i=1}^{N_j} \sum_{p=1}^n w_{ij} \|x_{ip} - c_{jp}\|^2 \quad (13-7)$$

$$c_{jp} = \frac{\sum_{i=1}^{N_j} w_{ij} x_{ip}}{\sum_{i=1}^{N_j} w_{ij}} \quad (j=1, \dots, M; p=1, \dots, n) \quad (13-8)$$

$$w_{ij} = \begin{cases} 1, & \text{若样品 } i \text{ 属于 } j \text{ 类} \\ 0, & \text{否则} \end{cases} \quad (j=1, \dots, M; i=1, \dots, N_j) \quad (13-9)$$

式中, x_{ip} 为第 i 个样品的第 p 个属性, c_{jp} 为第 j 个类中心的第 p 个属性。

5. 更新蚁群

在每一次蚁群更新中, 蚂蚁将通过信息素的间接通信实现把个 N 样品划分为 M 个类的一个近似划分。当 m 只蚂蚁都迭代结束后, 加入局部搜索以便进一步提高划分的质量, 然后根据划分的质量更新信息素矩阵。如此循环, 直到满足循环条件结束。

下面我们介绍具体的更新过程, 用 t 来表示迭代的次数。每只蚂蚁依赖于第 $t-1$ 次迭代提供的信息来实现分类。设表 13-2 表示本次迭代中继承的信息素矩阵。对每只蚂蚁所构成的每个样品, 系统产生一个随机数 q , 预先定义好概率 q_0 ($0 < q_0 < 1$), 对每只蚂蚁的更新根据概率 q_0 有两种方式:

① 若此随机数小于 q_0 , 则选择与样品间具有最大信息素的类为样品要归属的类。例如, 假设某次迭代中样品 1 归属于不同类的信息素为: $\tau_{11} = 0.31, \tau_{12} = 0.21, \tau_{13} = 0.41, \tau_{14} = 0.51$ 。这组数据表明, τ_{14} 的值较大, 当随机数 $q < q_0$ 时, 第 1 个样品将归属于第 4 类。

② 若此随机数大于 q_0 , 根据转换概率随机选择样品要转换的类。

计算样品 i 转换到类 j 的转换概率 p_{ij} :

$$p_{ij} = \frac{\tau_{ij}}{\sum_{l=1}^M \tau_{il}} \quad (j = 1, \dots, M) \quad (13-10)$$

这里 τ_{ij} 为样品 i 和所属类 j 间的标准化信息素。每个样品 i 根据转换概率分布, 选择要转换到的类别。

第一种方式为利用已有的知识, 而第二种方式为开发新解的空间。

6. 局部搜索

依照上述方法计算所有蚂蚁对应的解集。对每一只蚂蚁根据式 (13-7) ~ 式 (13-10) 计算其对应的聚类中心及其目标函数值。为提高算法中蚂蚁找到近似解的效率, 很多改进的蚁群算法都加入了局部搜索, 特别是当问题域的启发信息不易获得时, 加入局部搜索可以帮助找到更好的解。目前, 局部搜索可对所有解都执行, 也可以只对部分解执行。在局部搜索前, 把所有的解按照目标函数值进行升序排列。对具有小的目标函数值的前 L 个解进行局部搜索操作。局部搜索操作有很多种, 本算法选择变换操作。对目标函数值较小的前 L 个蚂蚁解集进行变换, 方法如下:

① 为解集中的每个样品产生随机数, 预先设定一个 $0 \sim 1$ 间的随机数 pls 。设只有第 i 个样品被分配的随机值小于 pls , 所以这个样品要被分到其他类当中。

② 选择类中心与这个样品的距离最短的类为第 i 个样品被分配的类, 重新聚类。

③ 根据式 (13-7) 重新计算变换操作后的目标函数值, 与原解集的目标函数值比较。若比原解集的目标函数值小, 保留新解集, 否则还原旧解集。

④ 对前 L 个解集进行上述操作。在前 L 个蚂蚁中选择具有最小目标函数的蚂蚁作为最优解。

7. 信息素矩阵更新

执行过局部搜索之后, 利用前 L 个蚂蚁对信息素表进行更新。信息素更新采用式 (13-11)

$$\tau_{ij}(t+1) = (1 - \rho)\tau_{ij}(t) + \sum_{s=1}^L \Delta\tau_{ij}^s \quad (i = 1, \dots, N; j = 1, \dots, M) \quad (13-11)$$

其中, $\rho(0 < \rho < 1)$ 为信息素蒸发参数, $\tau_{ij}(t)$ 为样品 i 与类 j 在 t 时刻的信息素浓度。设 J_s 为蚂蚁 s 目标函数值, Q 为一参数常量值。若蚂蚁 s 中的样品 i 属于 j 类, 则 $\Delta\tau_{ij}^s = Q/J_s$, 否则 $\Delta\tau_{ij}^s = 0$ 。

至此, 一次迭代结束。继续迭代, 直到达到最大迭代次数, 返回最优解为聚类结果。

8. 算法流程

最终找到的最优蚂蚁对应的解集如表 13-3 所示。如图 13-3 所示为聚类数目已知的蚁群聚类算法流程图。如图 13-4 所示为该解集对应的最优聚类划分。如图 13-5 所示为最终信息素矩阵。

表 13-3 最优蚂蚁的解集

	1	2	3	4	5	6	7	8	9	10	11	12
Sl	3	2	1	4	1	3	2	4	1	2	3	4

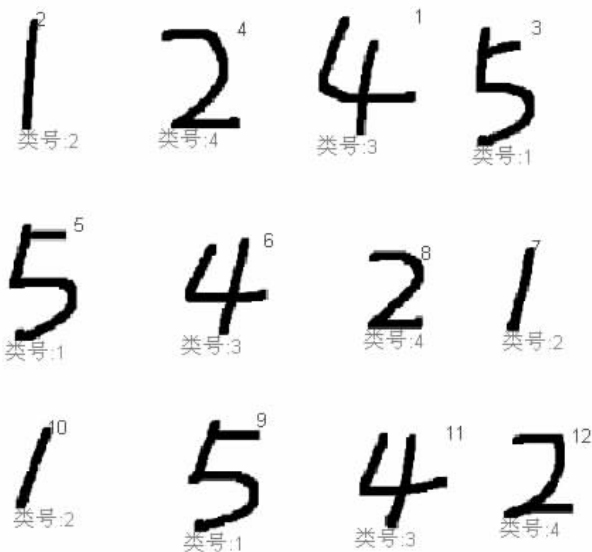
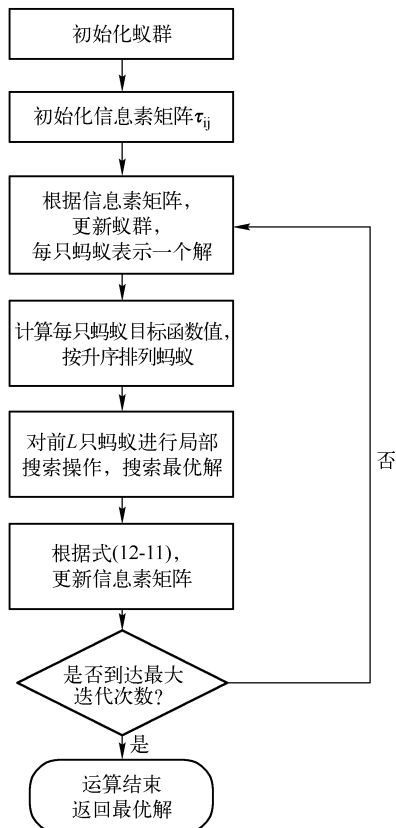


图 13-3 聚类数目已知的蚁群聚类算法流程图

图 13-4 蚁群算法找到的最优聚类划分

9. 实现步骤

① 初始化蚁群参数, 包括蚂蚁数目 antNum (取 200)、类的个数 centerNum 、转换规则参数 q (取 0.5)、信息素蒸发参数 ρ (取 0.1)、局部搜索阈值 (取 0.05) 等。

	1	2	3	4
1	0.0012	0.0012	0.0023	0.0012
2	0.0012	0.0022	0.0012	0.0013
3	0.0023	0.0012	0.0012	0.0012
4	0.0012	0.0012	0.0012	0.0024
5	0.0023	0.0012	0.0012	0.0012
6	0.0014	0.0013	0.0022	0.0012
7	0.0012	0.0023	0.0012	0.0012
8	0.0012	0.0014	0.0012	0.0022
9	0.0024	0.0012	0.0012	0.0012
10	0.0012	0.0024	0.0012	0.0012
11	0.0012	0.0012	0.0024	0.0012
12	0.0012	0.0012	0.0012	0.0023

图 13-5 最终信息素矩阵

- ② 初始化信息素矩阵。
- ③ 所有蚂蚁根据信息素矩阵构建解集。
- ④ 计算各类中心。计算每只蚂蚁的目标函数,并对蚂蚁按目标函数值排序。
- ⑤ 在排序后的蚂蚁解集中,将前 L 个蚂蚁作为要交换样品的蚂蚁,我们取 $L=2$,对要交换样品的蚂蚁实施局部搜索操作。
- ⑥ 按照式(13-11)更新信息素值。
- ⑦ 如果没有达到最大迭代次数,则转步骤③,否则输出最优聚类解集。

10. 编程代码

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 函数名称:C_AntClusterKnown()
% 参数:m_pattern:样品特征库;patternNum:样品数目
% 返回值:m_pattern:样品特征库
% 函数功能:按照蚁群聚类法对全体样品进行分类(已知聚类数目)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ m_pattern ] = C_AntClusterKnown( m_pattern,patternNum )
    disType = DisSelDlg();% 获得距离计算类型
    [ centerNum iterNum ] = InputClassDlg();% 获得类中心数和最大迭代次数
    % 初始化参数
    antNum = 200;% 蚂蚁数目
    rho = 0.1;% 1 - rho:信息素轨迹的衰减系数
    q = 0.5;% 转换规则参数
    tao = ones( patternNum,centerNum ) * 0.01;% 信息素矩阵初始化
    solution = zeros( antNum,patternNum + 1 );% 解矩阵初始化
    % 初始化聚类中心
    for i = 1:centerNum
        m_center(i).feature = m_pattern(i).feature;
        m_center(i).index = i;
        m_center(i).patternNum = 0;
    end
end

```

```

end
counter = 0;
bestVal = inf;
while( counter < iterNum)
    solution = 0 * solution;
    if( counter == 0)
        % 随机生成初始蚁群分布矩阵
        for i = 1 : antNum
            solution( i, 1 : patternNum) = randperm( patternNum) ;
            for j = 1 : patternNum
                if( solution( i, j) > centerNum)
                    solution( i, j) = fix( rand * centerNum + 1) ;
                end
                m _pattern( j). category = solution( i, j) ;
            end
            % 计算聚类中心
            for j = 1 : centerNum
                m _center( j) = CalCenter( m _center( j) , m _pattern, patternNum) ;
            end
            m _antCenter( i). m _center = m _center;
            % 计算目标函数
            for j = 1 : patternNum
                solution( i, patternNum + 1) = solution( i, patternNum + 1)
                    + GetDistance( m _pattern( j) , m _center( solution( i, j) ) , disType) ;
            end
        end
    end
else
    % 根据信息素矩阵计算每只蚂蚁的解
    r = rand( antNum, patternNum) ; % 产生随机数
    for i = 1 : antNum
        for j = 1 : patternNum
            if( r( i, j) < q) % 若 r < q, 选择与样品间具有最大信息素的类为样品要归属的类
                [ C, R] = max( tao( j, : ) ) ;
                solution( i, j) = R; % i 只蚂蚁的第 j 个样品对应 R 类
            else % 根据转换概率随机选择要归属的类
                temp = sum( tao( j, : ) ) ;
                tp = tao( j, : ) / temp; % 转移概率
                for m = 2 : centerNum
                    tp( m) = tp( m) + tp( m - 1) ;
                end
                rp = rand;
                for m = 1 : centerNum % 根据产生的随机数 r 判断归属的类
                    if( rp <= tp( m) )

```

```

        solution(i,j) = m;
        break;
    end
    end
    end
    m_pattern(j).category = solution(i,j);
end
% 计算聚类中心
for j = 1:centerNum
    m_center(j) = CalCenter(m_center(j), m_pattern, patternNum);
end
m_antCenter(i).m_center = m_center;
% 计算目标函数
for j = 1:patternNum
    solution(i, patternNum + 1) = solution(i, patternNum + 1)
        + GetDistance(m_pattern(j), m_center(solution(i,j)), disType);
end
end
end

% 对蚂蚁按目标函数由小到大排序
for i = 1:antNum - 1
    for j = i + 1:antNum
        if (solution(i, patternNum + 1) > solution(j, patternNum + 1))
            temp = solution(i, :);
            solution(i, :) = solution(j, :);
            solution(j, :) = temp;
            temp2 = m_antCenter(i);
            m_antCenter(i) = m_antCenter(j);
            m_antCenter(j) = temp2;
        end
    end
end
end
jp = 0.05; % 局部概率阈值
L = 2;
k = 1;
tempSolution = solution;
while (k <= L)
    r = rand(1, patternNum);
    for j = 1:patternNum
        m_pattern(j).category = solution(k, j);
    end
    for i = 1:patternNum

```

```

        if(r(1,i) < jp) % 该样品被分到其他类
            dis = inf;
            pj = 0;
            for j = 1:centerNum
                if(j ~ = m_pattern(i).category)
                    disTemp = GetDistance(m_pattern(i),
                        m_antCenter(k).m_center(j),disType);
                    if(dis > disTemp)
                        dis = disTemp;
                        pj = j;
                    end
                end
            end
            m_pattern(i).category = pj;
        end
    end
    % 计算调整后的的目标函数
    for j = 1:centerNum
        m_antCenter(k).m_center(j) = CalCenter(m_antCenter(k).m_center(j),m_pat-
            tern,patternNum);
    end
    solution(k,patternNum + 1) = 0;
    for j = 1:patternNum
        solution(k,patternNum + 1) = solution(k,patternNum + 1)
            + GetDistance(m_pattern(j),m_center(solution(k,j)),disType);
    end
    % 若调整后的目标函数小于先前的,保留调整后的样品所属的类
    if(solution(k,patternNum + 1) > tempSolution(k,patternNum + 1))
        solution(k,:) = tempSolution(k,:);
    end
    if(solution(k,patternNum + 1) < bestVal) % 保留最优解
        bestVal = solution(k,patternNum + 1);
        bestCluster = solution(k,:);
        t = counter;
    end
    k = k + 1;
end
if(bestVal == 0) % counter - t > iterNum/2 ||
    break;
end
% 信息素更新
tSum = zeros(patternNum,centerNum);
for m = 1:L

```

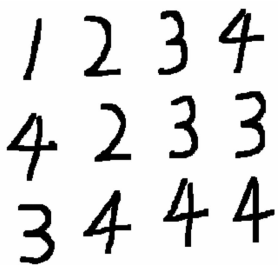
```

for i = 1:patternNum
    for j = 1:centerNum
        if( solution( m,i) == j)
            tSum(i,j) = tSum(i,j) + 0.001/solution( m,patternNum + 1 );
        end
    end
end
end
tao = ( 1 - rho) * tao + tSum;
counter = counter + 1;
end
for i = 1:patternNum
    m _pattern(i). category = bestCluster(i);
end
disp( tao );

```

11. 效果图

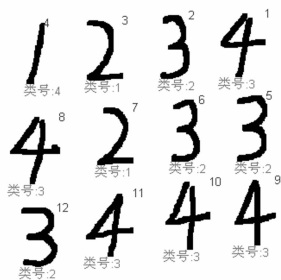
这里同样给读者提供两个实例效果图,一个是基于手写数字聚类结果图,如图 13-6 所示;另一个是基于图形聚类结果图,如图 13-7 所示。这两个图也比较复杂,但从结果可以看出,应用已知聚类数目的蚁群算法解决聚类问题效果非常好。注意:图右上角显示样品编号,左下角显示该样品所属类别。



(a) 原图数据



(b) 设定聚类数目和最大迭代次数



(c) 聚类结果

	1	2	3	4
1	0.0021	0.0015	0.0364	0.0133
2	0.0016	0.0468	0.0017	0.0021
3	0.0459	0.0012	0.0025	0.0023
4	0.0122	0.0068	0.0022	0.0316
5	0.0022	0.0475	0.0023	0.0014
6	0.0014	0.0469	0.0016	0.0016
7	0.0471	0.0019	0.002	0.0026
8	0.0017	0.0018	0.0445	0.0051
9	0.0014	0.0017	0.0439	0.006
10	0.0019	0.0016	0.0467	0.003
11	0.0017	0.0012	0.043	0.007
12	0.0016	0.0475	0.0018	0.0012

(d) 信息素(τ_{ij})矩阵

图 13-6 聚类数目已知的蚁群聚类算法效果图

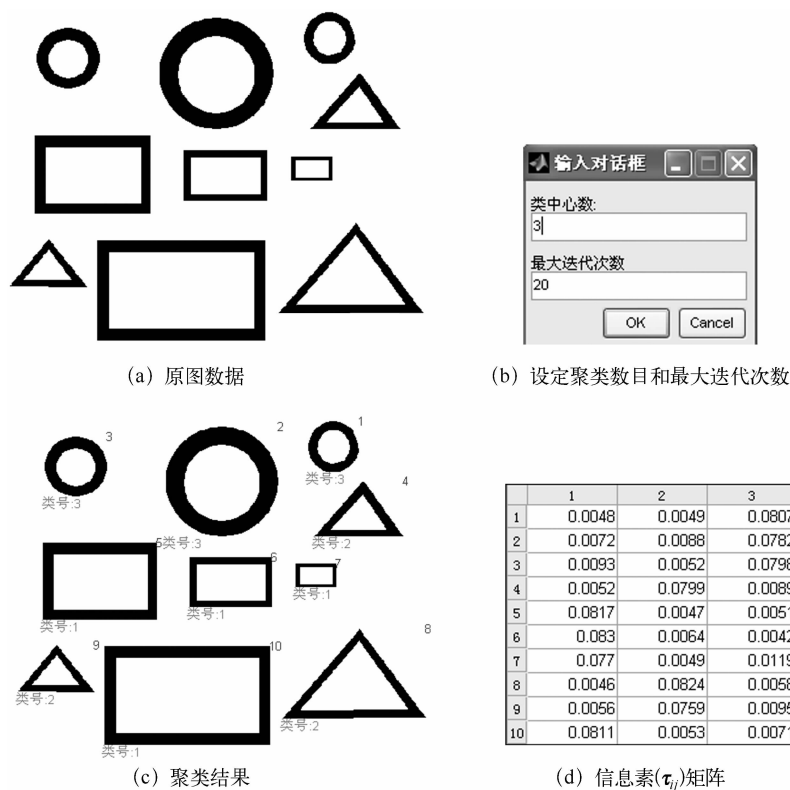


图 13-7 聚类数目已知的蚁群聚类算法效果图

13.3 聚类数目未知的蚁群聚类算法

1. 理论基础

在未知聚类数目的蚁群聚类算法,将样品视为具有不同属性的蚂蚁,聚类中心看做是蚂蚁要寻找的“食物源”。所以,样品聚类过程就看做是蚂蚁寻找食物源的过程。

设 $\mathbf{X} = \{X_i | X_i = (x_{i1}, x_{i2}, \dots, x_{in})\} (i = 1, 2, \dots, N)$ 是待进行聚类分析的数据集合, N 为样品的特征数,聚类过程可表述如下:

- ① 初始分配 N 个样品各自为一类 $(\omega_1, \omega_2, \dots, \omega_N)$ 。
- ② 计算类 ω_i 和 ω_j 之间的欧式距离 d_{ij} 。

$$d_{ij} = \|\overline{\mathbf{X}^{(\omega_i)}} - \overline{\mathbf{X}^{(\omega_j)}}\| = \sqrt{\sum_{k=1}^n (\overline{\mathbf{X}_k^{(\omega_i)}} - \overline{\mathbf{X}_k^{(\omega_j)}})^2} \quad (13-12)$$

$$\overline{\mathbf{X}^{(\omega_i)}} = \frac{1}{N_i} \sum_{k=1}^{N_i} \mathbf{X}_k (\mathbf{X}_k \in \omega_i) \quad (13-13)$$

式中, d_{ij} 表示 ω_i 到 ω_j 之间的欧式距离, $\overline{\mathbf{X}^{(\omega_i)}}$ 为聚类中心向量, N_i 为类 ω_i 中样品数量,可以根据各分量在聚类中的贡献不同而设定。

③ 计算各路径上的信息素量。设 r 为聚类半径, $\tau_{ij}(t)$ 是 t 时刻 ω_i 到 ω_j 路径上残留的信息素量,路径 (i, j) 上的信息素量定义为:

$$\tau_{ij}(t) = \begin{cases} 1, & d_{ij} \leq r \\ 0, & d_{ij} > r \end{cases} \quad (13-14)$$

式中, $r = A + d_{\min} + (d_{\max} - d_{\min}) \cdot B$, A, B 为常量参数, $d_{\min} = \min(d_{ij})$, $d_{\max} = \max(d_{ij})$ 。

④ 计算类 ω_i 归并到 ω_j 的概率:

$$p_{ij}(t) = \frac{\tau_{ij}^{\alpha}(t) \eta_{ij}^{\beta}(t)}{\sum_{s \in S} \tau_{is}^{\alpha}(t) \eta_{is}^{\beta}(t)} \quad (13-15)$$

式中, $S = \{s | d_{sj} \leq r, s = 1, 2, \dots, j-1, j+1, \dots, M\}$, s 代表某一类号, S 代表到第 j 类距离小于等于 r 的所有类号集合, M 为当前的总类数, $\eta_{is}(t)$ 为权重参数。

⑤ 判断若 $p_{ij}(t) \geq p_0$, 则 ω_i 归并到 ω_j 邻域, 类别数减 1。 p_0 为一给定的概率值。重新计算归并后类的聚类中心。

⑥ 判断是否有归并。若无归并, 则停止循环。否则, 转②继续迭代。

算法流程如图 13-8 所示。

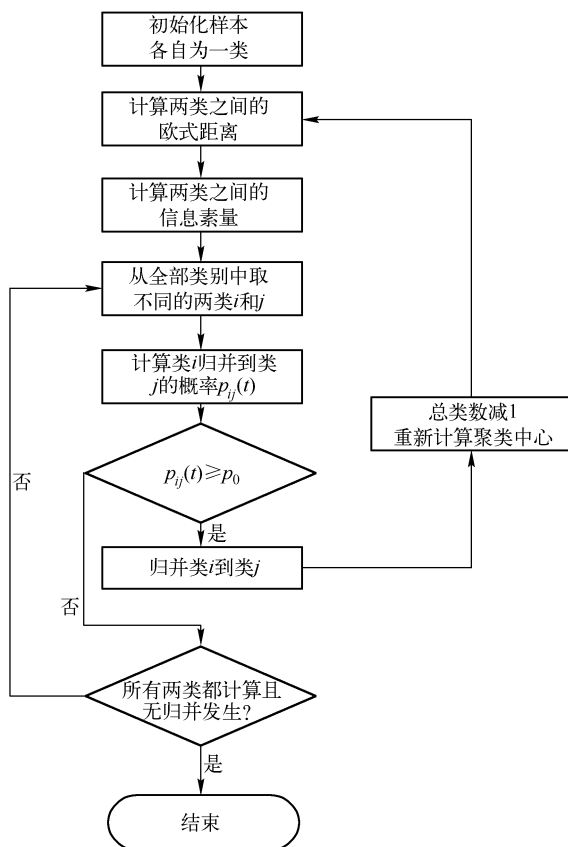


图 13-8 蚁群算法未知聚类数目算法流程

2. 实现步骤

① 初始化: 设定蚂蚁数量 N 为样品数量, n 为样品特征数目, 根据样品之间的最大(maxdis)、最小距离(mindis), 取 $r = 0.2 + \text{mindis} + (\text{maxdis} - \text{mindis}) \times 0.280$ 。 $\alpha = 1, \beta = 1$ 。 $\tau_{ij}(0) = 0$, 转移

概率 $p_0 = 0.65$ 。

- ② 按式(13-12),计算两类之间的距离。
- ③ 按式(13-14),计算各路径上的信息量。
- ④ 按式(13-15),计算 C_i 归并到 C_j 的概率。
- ⑤ 判断 $p_{ij}(t) > p_0$ 是否成立。若成立,将 ω_i 类归并到类 ω_j 。
- ⑥ 重新计算聚类中心。
- ⑦ 判断是否有归并。若无归并,则停止循环。否则,转②继续迭代。

3. 编程代码

```
%函数名称:C_AntCluster()
%参数:m_pattern;样品特征库;patternNum;样品数目
%返回值:m_pattern;样品特征库
%函数功能:按照未知聚类数同的蚁群聚类法对全体样品进行分类
function [ m_pattern ] = C_AntCluster( m_pattern, patternNum )
    disType = DisSelDlg(); % 获得距离计算类型
    m_min = inf;
    m_max = 0;
    % 计算最大最小距离
    for i = 1 : patternNum - 1
        for j = i + 1 : patternNum
            temp = GetDistance( m_pattern(i), m_pattern(j), disType );
            if( m_min > temp )
                m_min = temp;
            end
            if( m_max < temp )
                m_max = temp;
            end
        end
    end
    % 初始化参数
    r = m_min + ( m_max - m_min ) / 2;
    counter = 1;
    a = 1;
    b = 1;
    pc = 0.8;
    % 各个元素自成一类
    for i = 1 : patternNum
        m_center(i).index = i;
        m_pattern(i).category = i;
```

```

        m_center(i).feature = m_pattern(i).feature;
        m_center(i).patternNum = 1;
    end
    centerNum = patternNum;
    % 初始化参数
    tao = zeros(patternNum, patternNum);
    h = ones(patternNum, patternNum);
    distance = ones(patternNum, patternNum);
    p = zeros(patternNum, patternNum);
    temp = zeros(1, centerNum);
    while(counter < 100 && centerNum ~ = 1)
        % 计算类间距离
        for i = 1:centerNum - 1
            temp(i) = 0;
            for j = i + 1:centerNum
                distance(i, j) = GetDistance(m_center(i), m_center(j), disType);
                if(distance(i, j) < r)
                    tao(i, j) = 1;
                else
                    tao(i, j) = 0;
                end
                h(i, j) = 1;
                temp(i) = temp(i) + (tao(i, j)^a) * (h(i, j)^b);
            end
        end
        flag = 1;
        for i = 1:centerNum - 1
            for j = i + 1:centerNum
                if(temp(i) ~ = 0)
                    p(i, j) = (tao(i, j)^a) * (h(i, j)^b) / temp(i);
                else
                    p(i, j) = 0;
                end
            end
            if(p(i, j) > pc) % 归并 j 类到 i 类
                pi = i;
                pj = j;
                flag = 0;
                break;
            end
        end
        if(flag == 0)
            break;
        end
    end
end

```

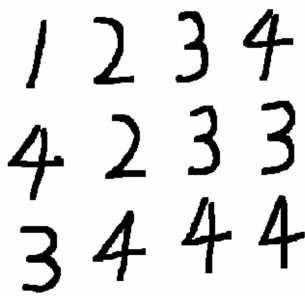
```

end
% 归并类
if(flag == 0)
    for i = 1:patternNum
        if(m_pattern(i).category == pj)
            m_pattern(i).category = pi;
        elseif(m_pattern(i).category > pj)
            m_pattern(i).category = m_pattern(i).category - 1;
        end
    end
    end
    centerNum = centerNum - 1;
else% 无归并时退出
    break;
end
% 重新计算聚类中心
for i = 1:centerNum
    m_center(i) = CalCenter( m_center(i),m_pattern,patternNum )
end
end
end

```

4. 效果图

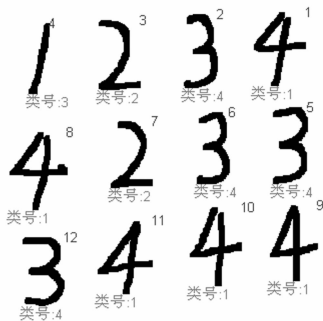
如图 13-9 所示为蚁群算法未知聚类数目聚类效果图。



(a) 原始数据



(b) 选择欧氏距离



(c) 聚类效果

图 13-9 蚁群算法未知聚类数目聚类效果图

本章小结

本章介绍了蚁群算法基本概念,包括蚁群算法的基本思想、蚁群算法的基本模型及特点;着重介绍了蚁群算法用于聚类问题的实现方法,包括聚类数目未知的蚁群聚类算法和聚类数目已知的蚁群聚类算法的实现方法和步骤。

习题 13

1. 简述蚁群算法的基本模型。
2. 简述蚁群算法进行旅行商问题求解步骤。
3. 对于聚类数目已知的蚁群算法聚类问题设计中,如果一幅位图当中包含有 15 个样品,分成 5 类,叙述蚂蚁的编码方法,给出一种可能的蚂蚁编码方案,蚂蚁长度应该设为多长? 如何计算蚂蚁的适应度? 信息素矩阵如何构造?
4. 叙述蚁群算法在已知聚类中心数目的聚类问题中的实现方法和步骤。
5. 叙述蚁群算法在未知聚类中心数目的聚类问题中的实现方法和步骤。

第 14 章 粒子群算法聚类分析

本章要点:

☑ 粒子群算法的基本概念

☑ 基于粒子群算法的聚类分析

14.1 粒子群算法的基本原理

粒子群算法(Particle Swarm Optimization, PSO)是一种有效的全局寻优算法,最早由美国的 Kennedy 和 Eberhart 于 1995 年提出。它是基于群体智能理论的优化算法,通过群体中粒子间的合作与竞争产生的群体智能指导优化搜索。与传统的进化算法相比,粒子群算法保留了基于种群的全局搜索策略,但是其采用的速度-位移模型,操作简单,避免了复杂的遗传操作,它特有的记忆使其可以动态跟踪当前的搜索情况调整其搜索策略。由于每代种群中的解具有“自我”学习提高和向“他人”学习的双重优点,从而能在较少的迭代次数内找到最优解。目前已广泛应用于函数优化、数据挖掘、神经网络训练等应用领域。

1. 粒子群算法的基本原理

粒子群优化算法具有进化计算和群智能的特点。起初 Kennedy 和 Eberhart 只是设想模拟鸟群觅食的过程,后来从这种模型中得到启示,并将粒子群算法用于解决优化问题。与其他进化算法相类似,粒子群算法也是通过个体间的协作与竞争,实现复杂空间中最优解的搜索。

粒子群算法中,每一个优化问题的解看做搜索空间中的一只鸟,即“粒子”。首先生成初始种群,即在可行解空间中随机初始化一群粒子,每个粒子都为优化问题的一个可行解,并由目标函数为之确定一个适应度值。每个粒子都将在解空间中运动,并由运动速度决定其飞行方向和距离。通常粒子将追随当前的最优粒子在解空间中搜索。在每一次迭代中,粒子将跟踪两个“极值”来更新自己,一个是粒子本身找到的最优解,另一个是整个种群目前找到的最优解,这个极值即全局极值。

粒子群算法可描述为:设粒子群在一个 n 维空间中搜索,由 m 个粒子组成种群 $Z = \{Z_1, Z_2, \dots, Z_m\}$,其中的每个粒子所处的位置 $Z_i = \{z_{i1}, z_{i2}, \dots, z_{in}\}$ 都表示问题的一个解。粒子通过不断调整自己的位置 Z_i 来搜索新解。每个粒子都能记住自己搜索到的最好解,记做 p_{id} ,以及整个粒子群经历过的最好的位置,即目前搜索到的最优解,记做 p_{gd} 。此外每个粒子都有一个速度,记做 $V_i = \{v_{i1}, v_{i2}, \dots, v_{in}\}$,当两个最优解都找到后,每个粒子根据式(14-1)来更新自己的速度。

$$v_{id}(t+1) = wv_{id}(t) + \eta_1 \text{rand}() (p_{id} - z_{id}(t)) + \eta_2 \text{rand}() (p_{gd} - z_{id}(t)) \quad (14-1)$$

$$z_{id}(t+1) = z_{id}(t) + v_{id}(t+1) \quad (14-2)$$

式中, $v_{id}(t+1)$ 表示第 i 个粒子在 $t+1$ 次迭代中第 d 维上的速度, w 为惯性权重, η_1, η_2 为加速

常数, $\text{rand}()$ 为 $0 \sim 1$ 之间的随机数。此外,为使粒子速度不致过大,可设置速度上限, v_{\max} 即当式(14-1)中 $v_{id}(t+1) > v_{\max}$ 时, $v_{id}(t+1) = v_{\max}$; $v_{id}(t+1) < -v_{\max}$ 时, $v_{id}(t+1) = -v_{\max}$ 。

从式(14-1)和式(14-2)可以看出,粒子的移动方向由三部分决定:自己原有的速度 $v_{id}(t)$ 、与自己最佳经历的距离 $p_{id} - z_{id}(t)$ 、与群体最佳经历的距离 $p_{gd} - z_{id}(t)$,并分别由权重系数 w, η_1, η_2 决定其相对重要性。如图 14-1 所示为 3 种移动方向的加权求和示意图。

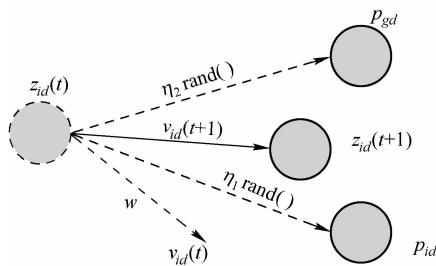


图 14-1 3 种移动方向的加权求和示意图

粒子群算法的基本流程如图 14-2 所示。

① 初始化粒子群,即随机设定各粒子的初始位置和初始速度 V_0 。

② 根据初始位置和速度产生各粒子新的位置。

③ 计算每个粒子的适应度值。

④ 对于每个粒子,比较它的适应度值和它经历过的最好位置 p_{id} 的适应度值,如果更好则更新。

⑤ 对于每个粒子,比较它的适应度值和群体所经历的最好位置 p_{gd} 的适应度值,如果更好则更新 p_{gd} 。

⑥ 根据式(14-1)和式(14-2)调整粒子的速度和位置。

⑦ 如果达到结束条件(足够好的位置或最大迭代次数),则结束,否则转步骤③继续迭代。

2. 全局模式与局部模式

Kennedy 等在对鸟群觅食的观察过程中发现,每只鸟并不总是能看到鸟群中其他所有鸟的位置和运动方向,而往往只是看到相邻的鸟的位置和运动方向。因此提出了两种粒子群算法模式:全局模式(global version PSO)和局部模式(local version PSO)。

全局模式是指每个粒子的运动轨迹受粒子群中所有粒子的状态影响,粒子追随两个极值,自身极值 p_{id} 和种群全局极值 p_{gd} ,式(14-1)和式(14-2)的算法描述的就是全局模式;而在局部模式中,粒子的轨迹只受自身的认知和邻近的粒子状态的影响,而不是被所有粒子的状态所影响,粒子除了追随自身极值 p_{id} 外,不追随全局极值 p_{gd} ,而是追随邻居粒子当中的局部极值 p_{nd} 。在该模式中,每个粒子需记录自己及其邻居的最优值,而不需要记录整个群体的最优值。此时,速度更新过程可用式(14-3)表示。

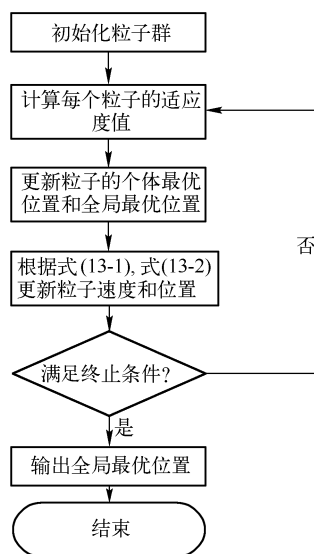


图 14-2 粒子群算法流程图示意图

$$v_{id}(t+1) = wv_{id}(t) + \eta_1 \text{rand}() (p_{id} - z_{id}(t)) + \eta_2 \text{rand}() (p_{nd} - z_{id}(t)) \quad (14-3)$$

全局模式具有较快的收敛速度,但是鲁棒性较差。相反,局部模式具有较高的鲁棒性而收敛速度相对较慢,因此在运用粒子群算法解决不同的优化问题时,应针对具体情况采用相应的模式。本章节所讨论的粒子群算法采用全局模式。

3. 参数选取

参数选取对算法的性能和效率有很大的影响。在粒子群算法中有 3 个重要参数:惯性权重 w 、速度调节参数 η_1 和 η_2 。惯性权重 w 使粒子保持运动惯性,速度调节参数 η_1 和 η_2 ,表示粒子向 p_{id} 和 p_{gd} 位置的加速项权重。如果 $w=0$,则粒子速度没有记忆性,粒子群将收缩到当前的全局最优位置,失去搜索更优解的能力。如果 $\eta_1=0$,则粒子失去“认知”能力,只具有“社会”性,粒子群收敛速度会更快,但是容易陷入局部极值。如果 $\eta_2=0$,则粒子只具有“认知”能力,而不具有“社会”性,等价于多个粒子独立搜索,因此很难得到最优解。

实践证明没有绝对最优的参数,针对不同的问题选取合适的参数才能获得更好的收敛速度和鲁棒性,一般情况下 w 取 0~1 之间的随机数, η_1, η_2 分别取 2。

4. 粒子群算法与其他进化算法的比较

(1) 相同点

粒子群算法与其他进化算法(如遗传算法和蚁群算法)有许多相似之处:

① 粒子群算法和其他进化算法都基于“种群”概念,用于表示一组解空间中的个体集合。它们都随机初始化种群,使用适应度值来评价个体,而且都根据适应度值来进行一定的随机搜索,并且不能保证一定能找到最优解。

② 种群进化过程中通过子代与父代竞争,若子代具有更好的适应度值,则子代将替换父代,因此都具有一定的选择机制。

③ 算法都具有并行性,即搜索过程是从一个解集合开始的,而不是从单个个体开始的,不容易陷入局部极小值。并且这种并行性易于在并行计算机上实现,提高算法的性能和效率。

(2) 不同点

粒子群算法与其他进化算法的区别:

① 粒子群算法在进化过程中同时记忆位置和速度信息,而遗传算法和蚁群算法通常只记忆位置信息。

② 粒子群算法的信息通信机制与其他进化算法不同。遗传算法中染色体互相通过交叉等操作进行通信,蚁群算法中每只蚂蚁以蚁群全体构成的信息素轨迹作为通信机制,因此整个种群比较均匀地向最优区域移动。在全局模式的粒子群算法中,只有全局最优粒子提供信息给其他的粒子,整个搜索更新过程是跟随当前最优解的过程,因此所有的粒子很可能更快地收敛于最优解。

14.2 基于粒子群算法的聚类分析

1. 理论基础

设模式样品集为 $X = \{X_i, i = 1, 2, \dots, N\}$, 其中 X_i 为 n 维模式向量, 聚类问题就是要找到一个划分 $\omega = \{\omega_1, \omega_2, \dots, \omega_M\}$, 使得总的类内离散度和达到最小。

$$J = \sum_{j=1}^M \sum_{X_i \in \omega_j} d(X_i, \overline{X^{(\omega_j)}}) \quad (14-4)$$

式中, $\overline{X^{(\omega_j)}}$ 为第 j 个聚类的中心, $d(X_i, \overline{X^{(\omega_j)}})$ 为样品到对应聚类中心距离, 聚类准则函数 J 即为各类样品到对应聚类中心距离的总和。

当聚类中心确定时, 聚类的划分可由最近邻法则决定。即对样品 X_i , 若第 j 类的聚类中心 $\overline{X^{(\omega_j)}}$ 满足式(14-5), 则 X_i 属于类 j 。

$$d(X_i, \overline{X^{(\omega_j)}}) = \min_{l=1,2,\dots,M} d(X_i, \overline{X^{(\omega_l)}}) \quad (14-5)$$

在粒子群算法求解聚类问题中, 每个粒子作为一个可行解组成粒子群(即解集)。根据解的含义不同, 通常可以分为两种方法: 一种是以聚类结果为解; 一种是以聚类中心集合为解。本节讨论的方法采用的是基于聚类中心集合作为粒子对应解, 也就是每个粒子的位置是由 M 个聚类中心组成, M 为已知的聚类数目。

一个具有 M 个聚类中心, 样品向量维数为 n 的聚类问题中, 每个粒子 i 由三部分组成, 即粒子位置、速度和适应度值。粒子结构 i 表示为

$$\text{Particle}(i) = \begin{cases} \text{location}[], \\ \text{velocity}[], \\ \text{fitness} \end{cases} \quad (14-6)$$

粒子的位置编码结构表示为

$$\text{Particle}(i). \text{location}[] = [\overline{X^{(\omega_1)}}, \overline{X^{(\omega_2)}}, \dots, \overline{X^{(\omega_M)}}] \quad (14-7)$$

式中, $\overline{X^{(\omega_j)}}$ 表示第 j 类的聚类中心, 是一个 n 维矢量。同时每个粒子还有一个速度, 其编码结构为

$$\text{Particle}(i). \text{velocity}[] = [V_1, V_2, \dots, V_M] \quad (14-8)$$

V_j 表示第 j 个聚类中心的速度值, 可知 V_j 也是一个 n 维矢量。

粒子适应度值 $\text{Particle}. \text{fitness}$ 为一实数, 表示粒子的适应度, 可以采用以下方法计算其适应度。

- ① 按照最近邻法式(14-5), 确定该粒子的聚类划分。
- ② 根据聚类划分, 重新计算聚类中心, 按照式(14-4)计算总的类内离散度 J 。
- ③ 粒子的适应度可表示为式(14-9)。

$$\text{Particle}. \text{fitness} = k/J \quad (14-9)$$

式中, J 为总的类内离散度和, k 为常数, 根据具体情况而定。即粒子所代表的聚类划分的总类

间离散度越小,粒子的适应度越大。

此外,每个粒子在进化过程中还记忆一个个体最优解 P_{id} ,表示该粒子经历的最优位置和适应度值。整个粒子群存在一个全局最优解 P_{gd} ,表示粒子群经历的最优位置和适应度。

$$P_{id}(i) = \left\{ \begin{array}{l} \text{location}[], \\ \text{fitness} \end{array} \right\} \quad (14-10)$$

$$P_{gd} = \left\{ \begin{array}{l} \text{location}[], \\ \text{fitness} \end{array} \right\} \quad (14-11)$$

根据式(14-1)和式(14-2),可以得到粒子 i 的速度和位置更新公式。

$$\begin{aligned} \text{Particle}(i), \text{velocil}[]' &= w\text{Particle}(i), \text{veiocity}[] \\ &+ \eta_1 \text{rand}() (P_{id}(i). \text{location})[] - \text{Particle}(i), \text{location}[] \\ &+ \eta_2 \text{rand}() (P_{gd}. \text{location})[] - \text{Particle}(i), \text{location}[] \end{aligned} \quad (14-12)$$

$$\text{Particle}(i). \text{location}[]' = \text{Particle}(i). \text{location}[] + \text{Particle}(i), \text{velocity}[]' \quad (14-13)$$

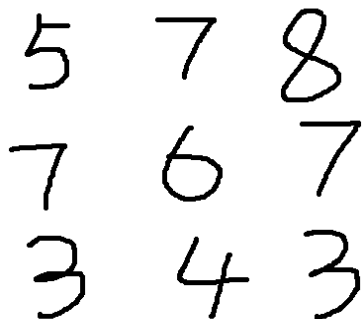


图 14-3 原始数据

根据已定义好的粒子群结构,采用上节介绍的粒子群优化算法,可实现求解聚类问题的最优解。

以如图 14-3 所示的原始数据为例,介绍粒子结构。从图上可见样品分为 6 个类别。首先产生 m 个粒子,每个粒子对图 14-3 中的样品随机分类,并计算各个类的中心 $\overline{X^{(\omega_i)}}$,初始速度 V_i 为 0。粒子编码示意图如表 14-1 所示,个体最优解 P_{id} 编码如表 14-2 所示,全局最优解 P_{gd} 编码如表 14-3 所示。

粒子群聚类算法流程如图 14-4 所示。

表 14-1 粒子编码

类中心 ω	$\overline{X^{(\omega_1)}}$	$\overline{X^{(\omega_2)}}$	$\overline{X^{(\omega_3)}}$	$\overline{X^{(\omega_4)}}$	$\overline{X^{(\omega_5)}}$	$\overline{X^{(\omega_6)}}$
每个类中心速度 V	V_1	V_2	V_3	V_4	V_5	V_6
适应度	$fitness = k/J = k/\sum_{j=1}^M \sum_{X_i \in \omega_j} d(X_i, \overline{X^{(\omega_j)}})$					

表 14-2 一个个体最优解 P_{id} 编码

粒子 i 的最优解 P_{id}	$\overline{X^{(\omega_1)}}$	$\overline{X^{(\omega_2)}}$	$\overline{X^{(\omega_3)}}$	$\overline{X^{(\omega_4)}}$	$\overline{X^{(\omega_5)}}$	$\overline{X^{(\omega_6)}}$
----------------------	-----------------------------	-----------------------------	-----------------------------	-----------------------------	-----------------------------	-----------------------------

表 14-3 全局最优解 P_{gd} 编码

全局最优解 P_{gd}	$\overline{X^{(\omega_1)}}$	$\overline{X^{(\omega_2)}}$	$\overline{X^{(\omega_3)}}$	$\overline{X^{(\omega_4)}}$	$\overline{X^{(\omega_5)}}$	$\overline{X^{(\omega_6)}}$
----------------	-----------------------------	-----------------------------	-----------------------------	-----------------------------	-----------------------------	-----------------------------

2. 实现步骤

① 粒子群的初始化。初始化粒子群,给定聚类数目 M ,粒子数量 m ,对于第 i 粒子 $\text{Particle}(i)$,先将每个样品随机指派为某一类,作为最初的聚类划分,并计算各类的聚类中心,作为粒子 i 的位置编码 $\text{Particle}(i). \text{location}[]$,计算粒子的适应度 $\text{Particle}(i). \text{fitness}$,设置粒子 i 的各中心的初始速度为 0。反复进行,生成 m 个粒子。

② 根据初始粒子群得到粒子的个体最优位置 $P_{id}(i)$ ($i=1, \dots, m$) 和全局最优位置 P_{gd} 。

③ 根据式(14-12)和式(14-13)更新所有粒子的速度和位置。其中 $\eta_1=2, \eta_2=2, \omega$ 按式(13-14)取值,其中 iter 为当前迭代次数, itermax 为最大迭代次数, $w_{\max}=1, w_{\min}=0$ 。式(14-14)可描述为 w 在迭代过程中由 w_{\max} 减小到 w_{\min} 。

$$w = w_{\max} - \text{iter} \times \frac{w_{\max} - w_{\min}}{\text{itermax}} \quad (14-14)$$

④ 对每个样品,根据粒子的聚类中心编码,按照最近邻法则,来确定该样品的聚类划分。

⑤ 对每个粒子,按照相应聚类划分,计算新的聚类中心,更新粒子的适应度值。

⑥ 对每个粒子 i ,比较它的适应度值和它经历过的最好位置的适应度值 $P_{id}(i)$,如果更好,更新 $P_{id}(i)$ 。

⑦ 对每个粒子 i ,比较它的适应度值和群体所经历的最好位置 P_{gd} 的适应度值,如果更好,更新 P_{gd} 。

⑧ 如果达到结束条件(得到足够好的位置或最大迭代次数),则结束算法,输出全局最优解;否则转步骤③继续迭代。

3. 编程代码

```
% 函数名称: C_PSO()
% 函数名称: C_PSO()
```

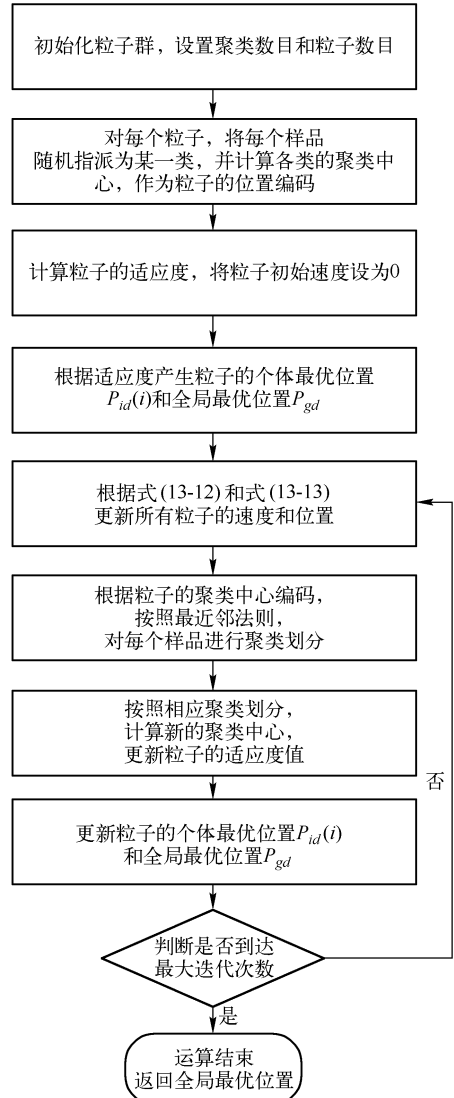


图 14-4 粒子群聚类算法流程示意图

```

% 参数:m_pattern:样品特征库;patternNum:样品数目
% 返回值:m_pattern:样品特征库
% 函数功能          按照粒子群聚类法对全体样品进行分类
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ m_pattern ] = C_PSO( m_pattern,patternNum )
    disType = DisSelDlg( );% 获得距离计算类型
    [ centerNum iterNum ] = InputClassDlg( );% 获得类中心数和最大迭代次数
    particleNum = 200;% 初始化粒子数目
    % 初始化中心和速度
    global Nwidth;
    for i = 1:centerNum
        m_center(i).feature = zeros( Nwidth,Nwidth );
        m_center(i).patternNum = 0;
        m_center(i).index = i;
        m_velocity(i).feature = zeros( Nwidth,Nwidth );
    for i = 1:particleNum
        Particle(i).location = m_center;% 粒子各中心
        Particle(i).velocity = m_velocity;% 粒子各中心速度
        Particle(i).fitness = 0;% 适应度
        P_id(i).location = m_center;% 粒子最优中心
        P_id(i).velocity = m_velocity;% 粒子最优速度
        P_id(i).fitness = 0;% 粒子最优适应度
    end
    P_gd.location = m_center;% 全局粒子最优中心
    P_gd.velocity = m_velocity;% 全局粒子最优速度
    P_gd.fitness = 0;% 粒子全局最优适应度
    P_gd.string = zeros( 1,patternNum );
    for i = 1:particleNum% 生成随机粒子分布矩阵
        ptDitrib( i,:) = ceil( centerNum. * rand( 1,patternNum ) );% 初始化类号
    end
    % 生成初始粒子群
    for i = 1:particleNum
        for j = 1:patternNum
            m_pattern(j).category = ptDitrib( i,j );
        end
        for j = 1:centerNum
            m_center(j) = CalCenter( m_center(j),m_pattern,patternNum );
        end
        Particle(i).location = m_center;
    end
End
% 初始化参数
w_max = 1;
w_min = 0;

```

```

h1 = 2;
h2 = 2;

for iter = 1 : iterNum
    % 计算粒子适应度
    for i = 1 : particleNum
        temp = 0;
        for j = 1 : patternNum
            temp = temp + GetDistance( m_pattern(j) , Particle(i). location
                ( ptDitrib(i,j) ) , disType );
        end
        if( temp == 0 ) % 最优解, 直接退出
            iter = iterNum + 1;
            break;
        end
        Particle(i). fitness = 1/temp;
    end
    if( iter > iterNum )
        break;
    end
    w = w_max - iter * ( w_max - w_min ) / iterNum; % 更新权重系数
    for i = 1 : particleNum % 更新 P_id, P_gd
        if( Particle(i). fitness > P_id(i). fitness )
            P_id(i). fitness = Particle(i). fitness;
            P_id(i). location = Particle(i). location;
            P_id(i). velocity = Particle(i). velocity;
            if( Particle(i). fitness > P_gd.fitness )
                P_gd.fitness = Particle(i). fitness;
                P_gd.location = Particle(i). location;
                P_gd.velocity = Particle(i). velocity;
                P_gd.string = ptDitrib(i, : );
            end
        end
    end
    % 更新粒子速度, 位置
    for i = 1 : particleNum
        for j = 1 : centerNum
            Particle(i). velocity(j). feature =
                w * Particle(i). velocity(j). feature + h1 * rand( Nwidth, Nwidth ). *
                ( P_id(i). location(j). feature - Particle(i). location(j). feature )
                + h2 * rand( Nwidth, Nwidth ). * ( P_gd.location(j). feature
                - Particle(i). location(j). feature );
            Particle(i). location(j). feature = Particle(i). location(j). feature

```

```

        + Particle(i). velocity(j). feature;
    end
end
% 最近邻聚类
for i = 1 : particleNum
    for j = 1 : patternNum
        min = inf;
        for k = 1 : centerNum
            tempDis = GetDistance( m _pattern(j) , Particle(i). location( k) ,
                disType );
            if( tempDis < min )
                min = tempDis;
                m _pattern(j). category = k;
                ptDitrib( i,j) = k;
            end
        end
    end
    % 重新计算聚类中心
    for j = 1 : centerNum
        Particle(i). location(j) = CalCenter( Particle(i). location(j) ,
            m _pattern, patternNum );
    end
end
for i = 1 : patternNum
    m _pattern(i). category = P _gd. string(1,i);
end
end
end

```

4. 效果图

如图 14-5 所示为粒子群聚类算法效果图。

本章小结

本章介绍了粒子群算法基本概念,包括粒子群算法的基本思想、粒子群算法的两种模型、参数选取及粒子群算法与其他进化算法的比较,着重介绍了粒子群算法用于聚类问题的实现方法和步骤。基于粒子群的聚类算法在产生下一代解时有较大的随机性,不容易陷入局部极小值,而且由于每代所有解“信息”的共享性和各个解的“自我素质”的提高,使得每代种群中的解具有“自我”学习提高和向“他人”学习的双重优点,所以具有较快的收敛速度。由于不存在随机寻优的退化现象,因此后期收敛比较平稳,很少出现波动现象。



图 14-5 粒子群聚类算法效果图

习题 14

1. 简述粒子群算法的基本原理。
2. 叙述粒子群算法与其他进化算法的异同。
3. 在粒子群算法聚类问题设计中,简述如何定义粒子结构以及粒子的更新方式。
4. 叙述粒子群算法在聚类问题中的实现方法和步骤。

参考文献

- [1] Beigy, H. ,Meybodi, M. R. . Adaptation of parameters of BP algorithm using learning automata. In Proceedings of Neural Networks. IEEE, 2000.
- [2] Wen Jin, Zhao Jia Li, Luo Si Wei, Han Zhen. The improvement of BP neuralnetwork learning algorithm. In Proceedings of Signal Processing. IEEE, 2000.
- [3] RoyChowdhury, P. ,Singh, Y. P. ,Chansarkar, R. A. . Dynamic tunneling technique for efficient training of multi-layer perceptrons[J]. Neural Networks, IEEE Transactions, 1999, 10(1) : 48-55.
- [4] Billings S A, Zheng G L. Radial basis function networks configuration using genetic algorithms[J]. Neural Networks, 1995, 8(6) : 877-890.
- [5] Chen, S. ,Cowan, C. F. N. ,Grant, P. M. . Orthogonal Least Square Learning Algorithm for Radial Basis Function Networks[J]. Neural Networks, IEEE Transactions, 1991, 2(2) : 302-309.
- [6] Kaminski, W. ,Strumillo, P. . Kernel Orthonormalization in Radial Basis Function Neural Networks[J]. Neural Networks, IEEE Transactions, 1997, 8(5) : 1177-1183.
- [7] Xu, L. ,Krzyzak, A. ,Oja, E. . Rival Penalized Competitive Learning for Clustering Analysis RBF Net and Curve Detection[J]. Neural Networks, IEEE Transactions, 1993, 4(4) : 636-649.
- [8] Pal, N. R. ,Bezdek, J. C. ,Tsao, E. C. - K. . Generalized Clustering Networks and Kohonen's Self- Organizing Scheme[J]. Neural Networks, IEEE Transactions, 1993, 4(4) : 549-557.
- [9] Plamondon Rejean, Srihari Sargur N. On-line and off-line handwriting recognition;a comprehensive survey[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(1) : 63-84.
- [10] Specht D F. Probabilistic neural networks[J]. Neural Networks, 1990, 3(1) : 109-118.
- [11] Streit, R. L. ,Luginbuhl, T. E. . Maximum likelihood training of probabilistic Neural networks[J]. Neural Networks, IEEE Transactions, 1994, 5(5) : 764-783.
- [12] Rustkowski L. A daptive probabilistic neural networks for pattern classification in time-varying environment [J]. Neural Networks, 2004, 15(4) : 811-827.
- [13] Z. B. Xu, C. P. Kwong. Global Convergence and Asymptotic Stability of Asymmetric Hopfield Neural Networks [J]. Mathematical Analysis and Applications, 1995, 191(3) : 405-427.
- [14] Kim, J. H. , Yoon, S. H. , Kim, Y. H. , Park, E. H. , Ntuen, C. , Sohn, K. H. , Alexander, W. E. . An efficient matching algorithm by a hybrid Hopfield network for object recognition. In Proceedings of Circuits and Systems. IEEE, 1994.
- [15] Abu- Mostafa, Y. , St. Jacques, J. . Information capacity of the Hopfield model[J]. Neural Networks, IEEE Transactions, 1985, 31(4) : 461-464.
- [16] Chen, L. -C. , Fan, J. -Y. , Chen, Y. -S. . A high speed modified Hopfield neural network and a design of character recognition system. In Proceedings of Security Technology, IEEE International Carnahan. IEEE, 1991.
- [17] Galan- Marin, G. , Munoz- Perez, J. . Design and Analysis of Maximum Hopfield Networks[J]. Neural Networks, IEEE Transactions, 2001, 12(2) : 329-339.
- [18] Li, S. Z. . Improving convergence and solution quality of Hopfield- type neural networks with augmented Lagrange multipliers. In Proceedings of Neural Networks. IEEE, 1996.

-
- [19] Abu- Mostafa, Y. , St. Jacques, J. . Information capacity of the Hopfield model[J]. Information Theory, IEEE Transactions, 1985, 31(4) : 461-464.
- [20] Galan- Marin, G. , Munoz- Perez, J. . Design and Analysis of Maximum Hopfield Networks[J]. Neural Networks, IEEE Transactions, 2001, 12(2) : 329-339.
- [21] Zheng Pei, Keyun Qin, Yang Xu. Dynamic adaptive fuzzy neural-network identification and its application. In Proceedings of Systems, Man and Cybernetics. IEEE, 2003.
- [22] Pan Zeng. Neural computing in mechanics[J]. Applied Mechanics Reviews, 1998, 51(2) : 173-197.
- [23] Khan, F. , Cervantes, A. . Real time object recognition for teaching neural networks. In Proceedings of Frontiers in Education. IEEE, 1999.
- [24] DORIGO M, GAMBARDELLA L M. A study of some properties of Ant-Q. In Proceedings of the 44th International Conference on Parallel Problem Solving from Nature. Springer- Verlag, 1996.
- [25] T. Krink, J. S. Vesterstrom, J. Riget. Particle swarm optimization with spatial particle extension. In Proceedings of IEEE on Evolutionary Computation. IEEE, 2002.
- [26] Xiaohui Hu, Eberhart, R. C. . Adaptive particle swarm optimization: detection and response to dynamic systems. In Proceedings of the 2002 Congress on Evolutionary Computation. IEEE, 2002.
- [27] Parsopoulos K E, Vrahatis M N. Recent approaches to global optimization problems through particle swarm optimization[J]. Natural Computing, 2002, 1(2-3) : 235-306.
- [28] Tin- Yau Kwok, Dit- Yan Yeung. A theoretically sound learning algorithm for constructive neural networks. In Proceedings of Speech, Image Processing and Neural Networks. IEEE, 1994.
- [29] Chandramouli, K. , Izquierdo, E. . Image Classification using Chaotic Particle Swarm Optimization. In Proceedings of Image Processing. IEEE, 2006.
- [30] Yang Shuying, He Peilian. Moving target detection through omni- orientational vision fixed on AGV. In Proceedings of SPIE- Intelligent Robots and Computer Vision XXIV : Algorithmtms, Techniques, and Active Vision. SPIE, 2006.
- [31] Yang Shuying, He Peilian. Design of Real- time Multi- targets Recognition System. In Proceedings of SPIE- MIPPR 2005: Image Analysis Techniques. SPIE, 2005.
- [32] Shu Ying Yang, Cheng Zhang, Wei Yu Zhang, Pi Lian He. Unknown Moving Target Detecting and Tracking Based on Computer Vision. In Proceedings of Fourth International Image and Graphics. IEEE, 2007.
- [33] 杨淑莹, 何丕廉. 基于遗传算法的目标识别实时系统设计[J]. 模式识别与人工智能, 2006(3) : 325-330.
- [34] 杨淑莹, 郭翠梨. FCCU 分流塔产品质量预测系统的设计[J]. 哈尔滨工业大学学报, 2005(4) : 501-503.
- [35] 杨淑莹, 王厚雪, 章慎锋, 何丕廉. 序列图像中运动目标聚类识别技术研究[J]. 天津师范大学学报自然科学版, 2005(3) : 51-53.
- [36] 杨淑莹, 王厚雪, 章慎锋. 基于图像分割的伪并行免疫遗传算法聚类设计[J]. 天津理工大学学报, 2006(5) : 85-87.
- [37] 杨淑莹, 王厚雪, 章慎锋. 基于 BP 神经网络的手写字符识别[J]. 天津理工大学学报, 2006, 22(4) : 82-84.
- [38] 杨淑莹. 基于机器视觉的齿轮产品外观缺陷检测[J]. 天津大学学报, 2007, 40(9) : 1111-1114.
- [39] 杨淑莹, 章慎锋, 王厚雪. 一种特定问题多目标识别系统设计[J]. 河北工业大学学报, 2005(3) : 105-108.
- [40] 杨淑莹, 王厚雪, 章慎锋. 基于 Bayes 决策的手写体数字识别[J]. 天津理工大学学报, 2006, 22(1) : 80-82.

- [41] 杨淑莹,郭翠梨. CLIPS 专家系统与神经网络 FCCU 分流塔装置的应用[J]. 计算机工程与应用,2005(3): 222-225.
- [42] 杨淑莹,王厚雪,章慎锋. 序列图像中多运动目标的识别[J]. 天津理工大学学报,2005(2): 3-5.
- [43] 杨淑莹,韩学东. 基于视觉的自引导车实时跟踪系统研究[J]. 哈尔滨工业大学学报,2004(11): 1471-1473.
- [44] 杨淑莹. 图像模式识别 VC++ 技术实现[M]. 北京:清华大学出版社,2005.
- [45] 杨淑莹. VC++ 图像处理程序设计[M]. 第2版. 北京:清华大学出版社,2004.
- [46] 张贤达. 矩阵分析与应用[M]. 北京:清华大学出版社,2003.
- [47] 张宏林. Visual C++ 数字图像模式识别技术及工程实践[M]. 北京:人民邮电出版社,2003.
- [48] 沈清,汤霖. 模式识别导论[M]. 长沙:国防科技大学出版社,1991.
- [49] 李月景. 图像识别技术及其应用[M]. 北京:机械工业出版社,1985.
- [50] 王碧泉,陈祖荫. 模式识别[M]. 北京:地震出版社,1989.
- [51] 边肇祺,张学工. 模式识别[M]. 北京:清华大学出版社,2000.
- [52] 边肇祺,张学工. 模式识别[M]. 第2版. 北京:清华大学出版社,2003.
- [53] 罗耀光,盛立东. 模式识别[M]. 北京:人民邮电出版社,1989.
- [54] Richard O. Duda, Peter E. Hart, David G. Stork. 模式分类[M]. 第2版. 北京:机械工业出版社,2003.
- [55] Sergios Theodoridis, Konstantinos Koutroumbas. 模式识别[M]. 第2版. 北京:机械工业出版社,2003.
- [56] 殷勤光,杨宗凯,谈正等编译. 模式识别与神经网络[M]. 北京:机械工业出版社,1992.
- [57] 高隽. 人工神经网络原理及仿真实例[M]. 北京:机械工业出版社,2003.
- [58] 史忠植. 神经计算[M]. 北京:工业出版社,1993.
- [59] 王旭,王宏,王文辉. 人工神经网络原理与应用[M]. 沈阳:东北大学出版社,2000.
- [60] 李敏强,寇纪淞,林丹,李书全[M]. 遗传算法的基本理论与应用[M]. 北京:科学出版社,2002.
- [61] 周冠雄. 计算机模式识别(统计方法)[M]. 武汉:华中工学院出版社,1986.
- [62] 戚飞虎,周源华,余松煜,郑志航等译. 模式识别与图像处理[M]. 上海:上海交通大学出版社,1989.
- [63] 黄振华,吴诚一. 模式识别原理[M]. 杭州:浙江大学出版社,1991.
- [64] 苏金明,阮沈勇. MATLAB 实用教程[M]. 北京:电子工业出版社,2006.
- [65] 徐东艳,孟晓刚. MATLAB 函数库查询辞典[M]. 北京:中国铁道出版社,2006.
- [66] 飞思科技产品研发中心编著. 神经网络理论与 BATLAB7 实现[M]. 北京:电子工业出版社,2006.
- [67] 梁循. 数据挖掘算法与应用[M]. 北京:北京大学出版社,2006.
- [68] 吴启迪,汪镭著. 智能蚁群算法及应用[M]. 上海:上海科技教育出版社,2004.
- [69] 李士勇. 蚁群算法及其应用[M]. 哈尔滨:哈尔滨工业大学出版社,2004.
- [70] Nello Cristianini John Shawe-Taylor. 支持向量机导论[M]. 北京:电子工业出版社,2005.
- [71] 雷英杰,张善文,李续武,周创明. MATLAB 遗传算法工具箱及应用[M]. 陕西:西安电子科技大学出版社,2005.
- [72] 高尚,杨静宇. 群智能算法及其应用[M]. 北京:中国水利水电出版社,2006.
- [73] 王岩,隋思涟,王爱青. 数理统计与 MATLAB 工程数据分析[M]. 北京:清华大学出版社,2006.
- [74] 陈念貽,钦佩,陈瑞亮,文聪. 模式识别方法在化工中的应用[M]. 北京:科学出版社,2002.
- [75] 李介谷,蔡国廉. 计算机模式识别技术[M]. 上海:上海交通大学出版社,1986.
- [76] 肖健华. 智能模式识别方法[M]. 广州:华南理工大学出版社,2006.
- [77] 数学手册编写组. 数学手册[M]. 北京:高等教育出版社,2004.
- [78] Sergios Theodoridis Konstantinos Koutroumbas. 模式识别[M]. 第二册. 北京:电子工业出版社,2004.
- [79] 唐启义,冯明光. DSP 数据处理身体——实验设计、统计分析及数据挖掘[M]. 北京:科学出版社,2007.

-
- [80] 董长虹. Matlab 神经网络与应用[M]. 北京: 国防工业出版社, 2005.
- [81] 范金城, 梅长林. 数据分析[M]. 北京: 科学出版社, 2004.
- [82] 陈仲生. 基于 Matlab7.0 的统计信息处理[M]. 长沙: 湖南科学技术出版社, 2005.
- [83] 求是科技编著. Matlab7.0 从入门到精通[M]. 北京: 人民邮电出版社, 2006.
- [84] 徐士良. C 常用算法程序集[M]. 北京: 清华大学出版社, 1996.
- [85] 王家文, 王皓, 刘海. Matlab7.0 编程基础[M]. 北京: 机械工业出版社, 2005.
- [86] 梁旭, 赵戈, 王民生. 改进的禁忌搜索算法求解多机并行模糊调度问题[J]. 大连交通大学学报, 2009(4): 51 - 54.
- [87] 梁旭, 黄明. 禁忌 - 并行遗传算法在作业车间调度中的应用[J]. 计算机集成制造系统 - CIMS, 2005(5): 678 - 681.
- [88] 威海英, 黄明, 李瑞. 一种求解 Job-Shop 调度问题的快速禁忌搜索算法[J]. 大连铁道学院学报, 2005(3): 46 - 48.
- [89] 黄明, 闫淑娟, 梁旭. 遗传算法和禁忌搜索算法在车间调度中的研究进展[J]. 工业控制计算机, 2004(2): 4 - 5.
- [90] 吴明光, 陈曦, 王明兴, 钱积新. 基于禁忌搜索算法的系统辨识[J]. 电路与系统学报, 2005(2): 108 - 111.
- [91] 董宗然, 周慧. 基于禁忌搜索算法的系统辨识[J]. 软件工程师, 2010(z1): 96 - 98.

反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可,复制、销售或通过信息网络传播本作品的行为;歪曲、篡改、剽窃本作品的行为,均违反《中华人民共和国著作权法》,其行为人应承担相应的民事责任和行政责任,构成犯罪的,将被依法追究刑事责任。

为了维护市场秩序,保护权利人的合法权益,我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为,本社将奖励举报有功人员,并保证举报人的信息不被泄露。

举报电话:(010)88254396;(010)88258888

传 真:(010)88254397

E-mail: dbqq@phei.com.cn

通信地址:北京市万寿路173信箱

电子工业出版社总编办公室

邮 编:100036